НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“ ” 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки ***6.050103 «Програмна інженерія»***

на тему *Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем*

Виконав: студент IV курсу, групи

ІП-51 Зарічковий Олександр Анатолійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ас. Ісаченко Г.В.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Рецензент:

доцент, к.т.н., доцент Марковський О.П. _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Зарічкового Олександра Анатолійовича
(прізвище, ім`я, по батькові)

1. Тема проекту « *Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем»*
керівник проекту *Ісаченко Георгій Вадимович, ас.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с
2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Розгортання та впровадження програмного забезпечення
4) Керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1) Структура генеративної мережі

2) Схема структурна класів програмного забезпечення,

3) Діаграма розгортання програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*20.04.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*20.04.2019*	
3.	*Постановка та формалізація задачі*	*25.08.2019*	
4.	*Аналіз вимог до програмного забезпечення*	*25.04.2019*	
5.	*Алгоритмізація задачі*	*28.04.2019*	
6.	*Моделювання програмного забезпечення*	*28.04.2019*	
7.	*Обґрунтування використовуваних технічних засобів*	*28.04.2019*	
8.	*Розробка архітектури програмного забезпечення*	*28.04.2019*	
9.	*Розробка програмного забезпечення*	*15.05.2019*	
10.	*Налагодження програми*	*19.05.2019*	
11.	*Виконання графічних документів*	*25.05.2019*	
12.	*Оформлення пояснювальної записки*	*25.05.2019*	
13.	*Подання ДП на попередній захист*	*28.05.2019*	
14.	*Подання ДП рецензенту*	*03.06.2019*	
15.	*Подання ДП на основний захист*	*08.06.2019*	

Студент _____ Зарічковий О.А.
(підпис)

Керівник проекту _____ Ісаченко Г.В.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проект	2	
2	A4	КПІ.ІП-5106.045490.01.81	Пояснювальна записка	62	
3	A4	КПІ.ІП-5106.045490.02.13	Опис програми	14	
4	A4	КПІ.ІП-5106.045490.03.91	Технічне завдання	10	
5	A4	КПІ.ІП-5106.045490.04.51	Програма та методика тестування	4	
6	A4	КПІ.ІП-5106.045490.05.34	Керівництво користувача	5	
7	A3	КПІ.ІП-5106.045490.06.99 Р	Структура генеративної мережі	1	
8	A3	КПІ.ІП-5106.045490.06.99 СС	Схема структурна класів програмного забезпечення	1	
9	A3	КПІ.ІП-5106.045490.06.99 СС	Діаграма розгортання програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5106.045490.00.90		
Розробн.		Зарічковий О.А.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Ісаченко Г.В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-51	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: “Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем ”

Київ – 2019 року

АНОТАЦІЯ

До бакалаврської дипломної роботи Зарічкового Олександра Анатолійовича на тему: «Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем».

Дипломна робота присвячена розробці алгоритму для деблюрінга зображень за допомогою генеративних змагальницьких систем та впровадженню його в використання через веб-застосунок. У роботі проведено порівняльний аналіз існуючих підходів для деблюрінга зображень на основі алгоритмів машинного навчання. Розроблені рекомендації щодо функціонального застосування розробленого програмного продукту.

Загальний обсяг роботи: 62 сторінки, 17 рисунків, 19 таблиць, 45 джерел.

ABSTRACT

Abstract to the bachelor thesis work of Zarichkovyi Oleksandr: "Software for image deblurring using generative adversarial systems."

This thesis is devoted to the development of an algorithm for the deblurring of images using generative competing systems and implemented as the web application. In this work was done a comparative analysis of existing approaches for the images deblurring that based on machine learning algorithms. Developed recommendations for the functional usage of the developed application.

Work includes 62 pages, 17 figures, 19 tables, 45 citations.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BPMN – система нотацій для моделювання процесів.

CycleGAN – алгоритм генерації зображень з двох різних розподілів без прямої відповідності між елементами даних розподілів.

DeblurGAN – алгоритм генерації чітких зображень з даних заблюрених зображень.

EM (Expectation Maximization) – алгоритм математичної статистики для знаходження оцінок максимальної правдоподібності параметрів ймовірнісних моделей від декількох змінних.

GAN – алгоритм генерації реалістичних зображень з заданого розподілу.

HTTP – протокол передачі інформації між інформаційними вузлами мережі.

HTTPS – захищена версія протоколу передачі даних HTTP.

REST – шаблон мережевих протоколів для забезпечення обміну інформацією між інформаційними вузлами в мережі.

ВСТУП

В останні 10 років відбувся великии прорив в областях машинного навчання, штучних неиронних мереж, і, як наслідок, у комп'ютерному зорі. Можливо наикращим прикладом цього буде проект "Asirra" від Microsoft Research[1]. Проект був засновании у 2006 році як засіб для відрізнення ботів від людеи в інтернеті. За иого задумкою користувачу ставилась задача відповісти кіт чи собака знаходиться на фотографіи. Зображення брались з постіино зростаючоі бази, що збиралася в приютах для тварин. Ця система безпеки вважалася маиже ідеальною, бо задача відрізнення кота від собаки вважалася AI-повною, тобто для іi вирішення необхідно створення повноцінного штучного інтелекту. У тои час найкращі алгоритми мали точність близько 60%, тоді як випадкове вадування має точність 50%. Тому имовірність що алгоритм зможе вірно підписати 15 зображень була мізерною (близько 0.047%), в тои час як для людини ця задача не складає жодних проблем. Результатом проекту стало иого закриття у 2014 році за причиною вирішення поставленоі задачі. Наикращі алгоритми на основі глибоких штучних неиронних мереж мали якість більше 99%, і, як наслідок, 15 зображень підписували вірно з имовірністю 86%, а 50 – з имовірністю 60.5% [2]. Кількість помилок у циіи надзвичаино складніи для комп'ютера задачі зменшилась у 40 разів та стала абсолютно мізерною. І все це лише за 8 років.

Звичаино, задача відрізнення котів від собак є далеко не наиважливішим здобутком машиного навчання та комп'ютерного зору. Є безліч куди важливіших та корисніших для людства задач: детекція об'єктів на зображеннях, розпізнання облич, розпізнання пішоходів і дорожніх знаків та багато інших. Але ця задача є дуже важливою з теоретичноі точки зору – вона показала фундаментальні проблеми та обмеження комп'ютерного зору, вирішення яких дуже сильно допомогло для розвитку інших напрямків.

На наш час більшість задач комп’ютерного зору, пов’язаних з розумінням та класифікацією того, що відбувається на зображеннях, мають хороше рішення, але з певними недоліками. На деяких подібних задачах комп’ютер навіть перевершив людину. Може скластися враження що комп’ютернии зір буде повністю вирішеною областю у найближчі роки. Але це не так.

Щодня в світі зі цифрових камер і мобільних телефонів знімається величезна кількість зображень, які поповнюють швидкоростучі бази цифрових фотографій [3] [4] [5] [6] [7]. Багато сучасних смартфонів оснащені фотокамерами з великою роздільною здатністю. Наприклад останній смартфон Mate 20 від компанії Huawei має камеру в 16 мегапікселів та здатний знімати фотографії з роздільною здатністю 4920×3264. Хоча сучасна фототехніка зробила значний прогрес, проте через ряд технічних проблем у більшості з них відсутні стабілізатори і як результат – вони не захищений тремтіння пристрою під час зйомки, що неминуче призводить до небажаного розмивання результуючих зображень. Хоча чіткі зображення можуть бути отримані за допомогою фіксації пристрою або повторної зйомки зображень, однак, у багатьох випадках ми не маємо можливості зафіксувати пристрій або повторно зняти зображення, наприклад, у космічних зондах [8], камерах відеоспостереження [9], медичних зображеннях [10] і деяких інших суміжних областях. Саме тому задачею деблюрінгу зображень займалися багато дослідників з різних галузей протягом багатьох років, але проблема все ще не може бути добре вирішена через складність виявлення джерела, що генерує розмиття та неоднозначності в інверсії даного процесу. Особливо гостро дані проблеми постають для реальних зображень, які містять велику кількістю дрібних деталей. Саме тому я обрав темою для даної роботи – деблюрінг зображень.

Мета цієї роботи – розробити алгоритм для деблюрінг зображень. Вирішення цієї задачі дозволило б покращити вже існуючі алгоритми обробки зображень та відео.

Найважливішою частиною роботи є підготовка даних та оцінка методу на них, що засвідчать про дієздатність розробленого підходу. При створенні програмного продукту пройдено повний цикл його написання – від постановки завдання, написання технічного завдання і вимог до продукту, до написання програмного продукту та його тестування.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Розмиття зображення – це певний процес деградації зображення, який виникає в результаті розфокусування камери при її руху або рухомими об’єктами на сцені. Приклад такої деградації наведено на рисунку 1.1. Зворотній процес до розмиття зображення називається деблюрінгом зображення.

Рисунок 1.1 – Розмиття зображення рухомими об’єктами

Математично даний процес може бути виражений наступним чином:

$$I_{blured} = F(I_{sharp}) + \varepsilon \quad (1.1),$$

де I_{sharp} – чітке зображення, I_{blured} – це розмита версія зображення I_{sharp}, F – функція деградації зображення, а ε – адитивний шум.

Оскільки загальний випадок, що описаний формулою 1.1, складний у розв’язанні через складність виявлення джерела, що генерує розмиття та неоднозначності в інверсії даного процесу, то у науковій літературі часто прибігають до аналізу часткового випадку, коли процес деградації є

рівномірним та інваріантним до зміщенням. Тоді процес, який описаний формулу 1.1, можна задати наступним чином:

$$I_{blured} = I_{sharp} * k + \varepsilon \text{ (1.2)},$$

де * - операція згортки, а k – ядро згортки, що породжує розмиття.

1.2 Змістовний опис і аналіз предметної області

1.2.1 Поняття зображення

У даній роботі під зображенням розуміється кольорове зображення у вигляді матриці цілих чисел від 0 до 255. Слід зауважити, що для більшості описаних алгоритмів кольорова модель або розмірність зображення не мають жодного значення.

1.2.2 Поняття операції згортки

У комп’ютерному зорі використовується особливий вид алгоритмів – згорточні неи□ронні мережі. Вони будуть мати дуже велике значення для нас у алгоритмі деблюрингу, тому необхідно зупинитися на операції згортки детальніше.

Згорточні неи□ронні мережі являються категорією неи□ронних мереж, які виявилися дуже ефективними в таких областях, як розпізнавання та класифікація зображень. Вони отримали своє ім'я від оператора згортки. Основною ідеєю згортки є те, що кількість параметрів у повнозв’язному шарі нейроної мережі можна дуже сильно зменшити, якщо прии□няти 2 гіпотези:

- функція кожного неи□рону залежить лише від невеликоі□ кількості неи□ронів попереднього шару, які відповідають за обробку тієї ж самої ділянки зображення;
- функція цієі□ залежності одна и□ та сама на всіх ділянках зображення.

Виходячи з вище сказаного можна дати наступне формальне визначення операції згортки. Згортка — це зазвичай невеликий просторовий (3-х вимірний)

тензор. Згортка отримала широке використання в комп'ютерному зорі оскіль за її допомогою можна ефективно виконувати операції розмиття зображення, підвищення різкості зображення, виділення границь об'єктів на зображенні та багато інших. Згортка зображення просторовим тензором полягає у обчисленні нового значень картинки на основі зваженої суми груп локальних пікселів на вхідному зображені.

Даваи□те розглянемо як це працює операція згортки на прикладі. Як вже говорилося вище, кожне зображення можна розглядати як матрицю значень пікселів. Розглянемо зображення 1.2, яке має розмірність 5 на 5 пікселів, які приймають значення лише 0 і 1. Також розглянемо аналогічне зображення 3 на 3 пікселя, яке зображено на рисунку 1.3.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Рисунок 1.2 – Оброблюване зображення з 0 та 1

1	0	1
0	1	0
1	0	1

Рисунок 1.3 – Матриця згортки

Тоді, згортка рисунку 1.2 матрицею згортки з рисунку 1.3 можна обчислити, як показано на рисунку 1.3:

Рисунок 1.4 – Обчислення операції згортки

Тобто, ми ковзаємо помаранчевою матрицею над нашим оригінальним зображенням, і для кожної позиції ми обчислюємо скалярний добуток між двома матрицями, щоб отримати остаточне число, яке утворює єдиний елемент вихідної матриці. Зауважимо, що кожний елемент вихідної матриці залежить лише від частини вхідної, у даному прикладу лише від квадрату 3 на 3 пікселі.

У термінології згорточних нейронних мереж матриця 3 × 3 називається фільтром або ядром, і вихід згорточного шару формується шляхом ковзання фільтра над вхідним зображенням. Різні значення матриці фільтра будуть створювати різні виходи згорточного шару для одного і того ж зображення. В якості прикладу розглянемо вхідне з рисунку 1.5.

Рисунок 1.5 – Вхідне зображення для згорточного шару

Подивимося, які ефекти згортки наведеного зображення ми можемо отримати за допомогою різних фільтрів:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Рисунок 1.6 – Фільтр та результат ідентичної функції

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Рисунок 1.7 – Фільтр та результат функції виділення країв

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Рисунок 1.8 – Фільтр та результат функції розмиття зображення

Отож, як ми бачимо, за допомогою згорток ми можемо робити досить складні перетворення зображень, та, що для нас є не менш важливо у задачі деблюрінга зображення – згортки зберігають просторову інформацію вхідного

зображення, а тому репрезентації згорточної нейронної мережі на внутрішніх шарах мають дуже інтуїтивно зрозумілий вигляд[6].

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Задача деблюрінга зображень являється класичною проблемою при обробці сигналів в системах, що працюють з зображенями. Умовно всі підходи для деблюрінга зображень можна розділити на два класи:

- параметричні;
- непараметричні.

Параметричні методи зазвичай вимагаю попереднього навчання на певному наборі даних для правильної оцінки всіх параметрі та вірного знаходження ядра згортки, що породжує розмиття, під час тестування методів. Непараметричні методи, на відміну від параметричних, не вимагають ніякого попереднього навчання та виконують оцінку ядра згортки розмиття на основі певних статистик зображення та максимізації апостеріорною правдоподібності. Оскільки більшість непараметричних методів являють собою певний набір ітеративні алгоритми, то їх швидкість в рази менша за швидкість параметричних аналогів, які виконуються за одну ітерацію. Історично склалося так, що до появи алгоритмів глибинного навчання, параметричні моделі значно уступали в якості непараметричним моделями, тому більшість ранніх робіт посвячена саме непараметричним моделям.

1.3.1.1 Непараметричні підходи

Більшість робіт, що вивчають непараметричні методи, які не потребують попереднього навчання, припускають, що процес розмиття зображень є інваріантним до здвигів і викликаний виключно рухом фотомодуля або об’єктів на зображенні [13] [14] [15], а сам процес може бути апроксимований та обернений за допомогою операції транспонованої згортки – деконволюції [13]

[16] [17] [18]. Для оцінки ваг ядра деконволюції багато робіт [13] використовували байєсівську оцінку наступного функціоналу, що наведено в формулі 1.3:

$$P(I_{sharp}, k \mid I_{blured}) \equiv P(I_{blured} \mid I_{sharp}, k)P(I_{sharp})P(k) \quad (1.3),$$

де I_{sharp} – різке зображення, I_{blured} – розмита версія зображення I_{sharp}, k – ядро згортки розмиття.

Різні непараметричні алгоритми використовують різні методи для оцінки різкого зображення та ядра згортки розмиття. В науковій номенклатурі найчастіше зустрічаються алгоритм, які основані на максимізації апостеріорної правдоподібності. Дані припускаючи незалежність ядра k та різкого зображення I_{sharp} і обчислюють ваги ядра згортки за допомогою апроксимації функціоналу, що описується формулою 1.4:

$$(k, I_{sharp}) = argmax_{k,I_{sharp}} P(I_{blured} \mid I_{sharp}, k)P(I_{sharp})P(k) \approx$$

$$\approx argmax_{k,I_{sharp}} \int p(I_{blured} \mid k)p(k)dI_{sharp} \quad (1.4)$$

Оскільки не існує аналітичного рішення для формули 1.4, а самі зображення I_{sharp} та I_{blured} дискретні, то на практиці рішення знаходиться за допомогою застосування алгоритму EM, який ітеративно параметри ядра згортки, що генерує розмите зображення.

1.3.1.2 Параметричні підходи

Параметричні підходи для деблюрінга зображень майже не зустрічалися у науковій номенклатурі до появи алгоритму DeblurGAN, який був представлений магістрами Українського Католицького університету у 2017 року [19]. Поява алгоритму DeblurGAN, що базуються на алгоритмах глибинного машинного навчання, дало значний скачок в якості деблюрінг зображень, що дозволило покращити результати на всіх основних еталонних наборах даних, які використовують для порівняння різних алгоритмів деблюрінга [19]. В основу DeblruGAN лягло дві ключові ідеї, які допомогли досягнути таких значних результатів:

- алгоритм генерації реалістичного розмиття для формування датасета, який можна використати для навчання алгоритмів з учителем;
- використання змагальницьких генеративних систем для генерації реалістичних результатів.

Давайте розглянемо ці два компоненти детальніше.

1.3.1.3 Алгоритм генерації тренувального набору даних для алгоритму DeblurGAN

Не існує легкого спосібу отримати пари зображень відповідних різких і розмитих зображень для навчання, на відміну від інших популярних проблем, що також генерують на виході картинку, таких як збільшення роздільної здатності зображень або розфарбування зображень. Типовий підхід до отримання пар зображень для навчання полягає у використанні фотокамери з великою швидкість зйомки зображень для імітації розмиття за допомогою усереднення сусідніх чітких кадрів з відео [20] [21]. Даний підхід дозволяє створювати реалістичні розмиті зображення, але обмежує кількість сцен кількістю знятих відео, що зменшує репрезентативність сформованого набору даних та ускладнює масштабування даних. Інші підходи базуються на використані лінійних ядер розмиття з різними траєкторіями, які генерують випадковим чином та застосовуються до зображень [22] [23]. Приклади таких траєкторій та результатів розмиття наведені на рисунку 1.9:

Рисунок 1.9 – Генерація розмиття на основі лінійних ядер за допомогою траєкторій розмиття. Зліва – оригінальне зображення. По центру – знервоване зображення з розмиттям. Справа – траєкторія генерації розмиття

В їх основі даних алгоритмів лежить ідея Борачі та Фоя [24] по генерації випадкових траєкторій. Ядра згорток розмиття генеруються шляхом застосування субпіксельної інтерполяції до вектора траєкторії розмиття. Кожен такий вектор є комплексним та відповідає дискретним положенням об'єкта після випадкового руху. Положення наступної точки траєкторії генерується випадковим чином на основі положення поточної точки, швидкості та імпульсу розмиття попередньої точки та траєкторії розмиття. Даний алгоритм узагальнений та представлений на рисунку 1.10:

Algorithm 1 Motion blur kernel generation. Initial parameters:
$M = 2000$ – number of iterations, $L_{max} = 60$ – max length of the movement,
$p_s = 0.001$ – probability of impulsive shake.
I = inertia term, uniform from (0,0.7)
p_b – probability of big shake, uniform from (0,0.2)
p_g – probability of gaussian shake, uniform from (0,0.7)
ϕ – initial angle, uniform from $(0,2\pi)$
x – trajectory vector

procedure BLUR(Img, M, L_{max}, p_s)
  $v_0 \leftarrow cos(\phi) + sin(\phi) * i$
  $v \leftarrow v_o * L_{max}/(M-1)$
  $x = \text{zeros}(M, 1)$
  for $t = 1$ to $M - 1$ **do**
    if randn $< p_b * p_s$ **then**
      nextDir $\leftarrow 2 \cdot v \cdot e^{i*(\pi+(\text{randn}-0.5))}$
    else:
      nextDir $\leftarrow 0$
    $dv \leftarrow \text{nextDir} + p_s * (p_g * (\text{randn} + i * \text{randn}) * I * x[t] * (L_{max}/(M-1))$
    $v \leftarrow v + dv$
    $v \leftarrow (v/abs(v)) * L_{max}/(M-1)$
    $x[t+1] \leftarrow x[t] + v$
  Kernel $\leftarrow$ sub pixel interpolation(x)
  Blurred image $\leftarrow conv(\text{Kernel}, \text{Img})$
  return Blurred image

Рисунок 1.10 – Лістинг генерації розмиття для зображення

1.3.1.4 Генеративні змагальницькі системи

Генеративні змагальницькі мережі являються молодою гілкою алгоритмів глибинного машинного навчання, які знайшли широке поширення в алгоритмах генерації контенту комп'ютерним зором та використовуються для навчання на нерозмічених наборах даних. Генеративні змагальницькі мережі реалізовані в вигляді системи що складається з двох або більше глибоких згорткових нейромереж, які змагаються однин з однин. Оскільки вони змагаються один з одним, то завдання кожної з мереж являється перевершення результатів її

суперників. Генеративні змагальниць мережі були вперше опубліковані інженером компанії Google – Яном Гудфелоу в 2014 році. [25] [26] Його революційний підхід дозволяє генерувати фотографії, які на перший погляд для людського ока виглядають як справжні.

В основу методу роботи генеративних змагальницьких мереж лягли два ключові блоки нейромережі: одні нейромережі генерують кандидатів (так званий генератори), а інша пробують відрізнити згенеровані зображання від реальних (так звані дискримінатори). Як правило, генеративні мережі навчаються знаходити певну відповідності між прихованим латентним простором та певним розподіл реальних даних. На відміну від генераторів дискримінатори навчаються розрізняти елементи справжнього розподілу даних від зфабрикованих елементі, які були вироблені генеративною нейромережею. Головною цілю генеративних нейромереж являється максимізація кількості помилок дискримінаційних неймомереж (або ж простішими словами – ціль «обдурити» дискримінатори шляхом створення синтетичних семплів з реального розподілу даних, які схожі на представників справжнього розподілу даних та не відлічимі від реальних даних за поліноміальний час), тоді як ціль дискримінаторів – їх мінімізація.

На практиці використовують певний фіксований набір даних як відправну точку в навчанні дискримінатора. Навчання дискримінатора відбувається на даному набору даних до тих пір поки він не досягне свого максимального рівня точності – так званого «плато». На противагу дискримінатору генератор на протягом свого навчання отримує випадково згенеровані елементи зі заздалегідь відомого латентного простору. На практиці часто використовують багатовимірний нормований нормальний розподіл як джерело елементів даного латентного простору які потрапляють на вхід в генераторну нейромережу. Тоді елементи, які згенеровані генератором, оцінюються дискримінатором. На навчання обох мереж використовується метод зворотного поширення помилки який реалізований на основі алгоритму стохастичного градієнтного спуску.

Даний підхід використовуються одночасно в обох мережах, так щоб генератор щоітерації створював все кращі та кращі зображення, тоді як дискримінатор умів відрізняти все складніші та складніші підробки від справжніх зображень. Генеративна нейромережа являє собою згорткову нейронну мережу з транспонованими згортками для збільшення просторової розмірності генерованих прикладів, а дискримінатор — звичайною згортковою нейронною мережею з мінімальною кількість розріджених шарів. Узагальнена схема генеративних змагальницьких мереж зображена на рисунку 1.11:

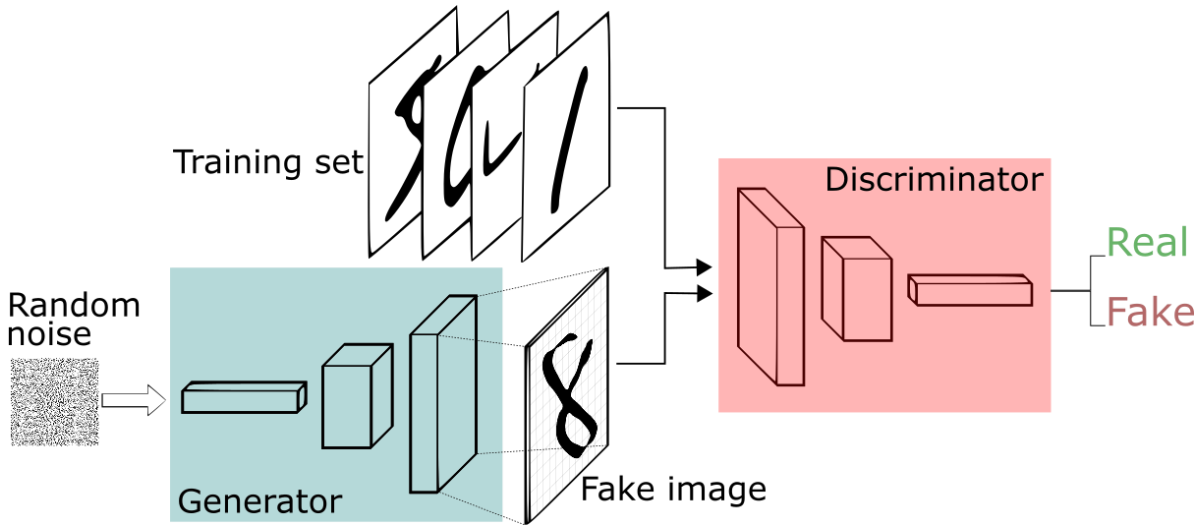


Рисунок 1.11 – Загальна схема генеративних змагальницьких систем запропонованих Яном Гудфелоу

Принцип змагальності в GAN нерідко описується через приклад з фальшивомонетчком. Генератор уподібнюється фальшивомонетнику або підроблювачу картин, а дискримінатор – експерту який прагне розпізнати підробку. Обидві мережі повинні щоразу удосконалюватися для того щоб перевершити навички суперника.

У популярному додатку генерації людських облич в якості справжніх даних виступають реальні фотографії, а генеративна мережа намагається створити штучні особи, варіюючи комбінації таких латентних параметрів, як колір волосся, пропорції обличчя, розріз очей, форма носа, розмір вух, наявність бороди і вусів і т.д.

1.3.1.5 DeblurGAN

Основна ідея DeblurGAN полягає в тому, щоб відновити чітке зображення для заданої розмитої версії того ж зображення без всяких обмежень на джерело, що генерує розмиття. Відновлення зображення здійснюється натренерованою згорточною нейрноною мережею, яка називається генератором. Також для кожного генератора тренеруєся окрема згорточна нейрнона мережа, так званий критик, який навчається відрізняти згнеровані зображення генератором від реальних зображень, а обидві мережі навчаються в змагальницькій формі шляхом мінімізації наступного функціоналу, що описаний формулою 1.5 [19]:

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda \cdot \mathcal{L}_X \quad (1.5)$$

Даний функціонал складається зі зваженої суми двох складових:

- змагальницького функції втрат;
- функція схожості контенту зображень.

Змагальницька функція втрат складається функції втрат генератора, що вимірює якість згенерованих зображень, та функції втрат критика, що вимірює наскільки критик здатен відрізняти реальні зображення від зображень згенерованих критиком. Змагальницька функція втрат описується формулою 1.6:

$$\mathcal{L}_{GAN} = \sum_{n=1}^{N} -D_{\theta_D}(G_{\theta_G}(I^B)) \quad (1.6)$$

Функція схожості контенту зображення вимірює перцептральну схожість згенерованого та цільового зображень і описується формулою 1.7:

$$\mathcal{L}_X = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^S)_{x,y} - \phi_{i,j}(G_{\theta_G}(I^B))_{x,y})^2 \quad (1.7)$$

Перцептральна схожість вимірюється за допомогою порівняння активацій внутрішніх шарів преднавчаної нейроної мережі VGG16.

Загальна архітектура DeblurGAN описана на рисунку 1.12. Архітектура генератора наведена в Додатку Г, Лист 1 Схема структурна генеративної мережі.

Рисунок 1.12 – Архітектура DeblurGAN

1.3.1.6 Недоліки DeblurGAN

Основними обмеженням DeblurGAN є те, що навчальний процес вимагає генерації попарного набору даних, який важко отримати в реальному світі. Незважаючи на те, що DeblurGAN надає метод генерації синтетичних розмитих зображень зображеннями із заданих чітких картинок, подальше використання фальшивих «розмитих» зображень є неперспективним, оскільки лімітує загальну потужність джерел розмиття лише лінійними ядрами. Тому відсутність навчального набору даних обмежує сферу застосування DeblurGAN. Саме на ліквідацію цього обмеження націлена дана робота.

1.3.2 Аналіз відомих програмних продуктів

Найбільш успішною комерційною реалізацією алгоритму деблюрингу зображень являється програмне забезпечення SmartDeblur [27]. В його основі лежить реалізація DeblurGAN, що навчається на більш загальному наборі даних аніж оригінально реалізація DebluGAN, проте алгоритм генерації розмиття для навчальних прикладів аналогічний тому, що представлений в DeblurGAN зі всіма його плюсами та мінусами, а отже його застосування дещо обмежений, оскільки більшість джерел в реальних кейсах мають не лінійні ядра [13] [16] [17] [18].

1.4 Аналіз вимог до програмного забезпечення

Виходячи з аналізу наявного функціоналу основних конкурентів, їх основних переваг та недоліків можна сформувати ряд функціональних та не функціональних вимог, які необхідні для створення конкурентного програмного забезпечення. Оскільки більшість конкурентів використовують десктопні додатки, які не потребують ніякої реєстрації, та зважаючи на законопроект GDPR, який прийняти в Євросоюзі та накладає суттєві обмеження на оброку чуттєвої інформації громадян Євросоюзу, то розроблюване програмне забезпечення також не буде мати функціоналу по авторизації

користувачів, оскільки дана інформація не потрібна для повноцінно функціонування веб-додатка з деблюрінгом зображення та спростить використання розробленого додатку в Європейському союзі, оскільки жодна інформація про користувачі не буде зберігатися на сайті, а отже неможливо ідентифікувати кому належить оброблювана інформація та будь-як використати дану інформацію.

1.4.1 Розроблення функціональних вимог

В системі передбачено наступні функціональні варіанти використання:

Таблиця 1.1 – Варіант використання UC001

Назва	Обробка оригінального зображення.
Опис	Користувач системи обробляє зображення.
Учасники	Користувач системи.
Передумови	
Постумови	Користувач отримав оброблене зображення.
Основний сценарій	1) Користувач відриває веб-додаток; 2) Користувач нажимає на кнопку «Завантаження зображення»; 3) Користувач обирає файл для обробки в файловій системі та нажимає на кнопку «Завантажити»; 4) Система відображає завантажене зображення; 5) Користувач нажимає на кнопку «Обробити». 6) Система виконує обробку завантаженого зображення та відображає оброблене зображення.

Продовження таблиці 1.1

Розширення сценаріїв	3.1) Система виявляє, що дані, обрані користувачем не являються зображенням. 3.1.а) Система демонструє користувачеві повідомлення про помилку завантаження файлу.

Таблиця 1.2 – Варіант використання UC002

Назва	Обробка зображення зі зміненим розміром.
Опис	Користувач системи обробляє зображення.
Учасники	Користувач системи.
Передумови	
Постумови	Користувач отримав оброблене зображення.
Основний сценарій	1) Користувач відриває веб-додаток; 2) Користувач нажимає на кнопку «Завантаження зображення»; 3) Користувач обирає файл для обробки в файловій системі та нажимає на кнопку «Завантажити»; 4) Система відображає завантажене зображення; 5) Користувач нажимає на випадаючому список «Розмір зображення»; 6) Система демонструє можливі варіанти вибору в випадаючому списку; 7) Користувач обирає підходящий варіант зі списку та нажимає на кнопку «Обробити». 8) Система виконує обробку завантаженого зображення та відображає оброблене зображення.

Продовження таблиці 1.2

Розширення сценаріїв	3.1) Система виявляє, що дані, обрані користувачем не являються зображенням. 3.1.а) Система демонструє користувачеві повідомлення про помилку завантаження файлу.

Виходячи з вище описаних вимого можна сформувати наступні функціональні вимоги:

Таблиця 1.3 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Завантаження зображення
Опис	Система має дозволяти користувачеві обрати файл в файловій системі та завантажити його для подальшої обробки системою.

Таблиця 1.4 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Відображення завантаженого зображення
Опис	Система має відобразити завантажене користувачем зображення.

Таблиця 1.5 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Зміна розміру оброблюваного зображення

Продовження таблиці 1.5

Опис	Система має надавати можливість змінити розмір завантаженого зображення перед тим як почати його обробку для пришвидшення процесу отримання фінального результату.

Таблиця 1.6 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Обробка завантаженого зображення
Опис	Система має виконує обробку завантаженого зображення

Таблиця 1.7 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Відображення обробленого зображення
Опис	Система має відобразити оброблене зображення користувачеві.

Таблиця 1.8 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Збереження обробленого зображення
Опис	Система має надавати користувачеві змогу зберегти оброблене зображення в файловій системі.

Залежності між функціональними вимогами клієнтського застосунку зображено на рисунку 1.13.

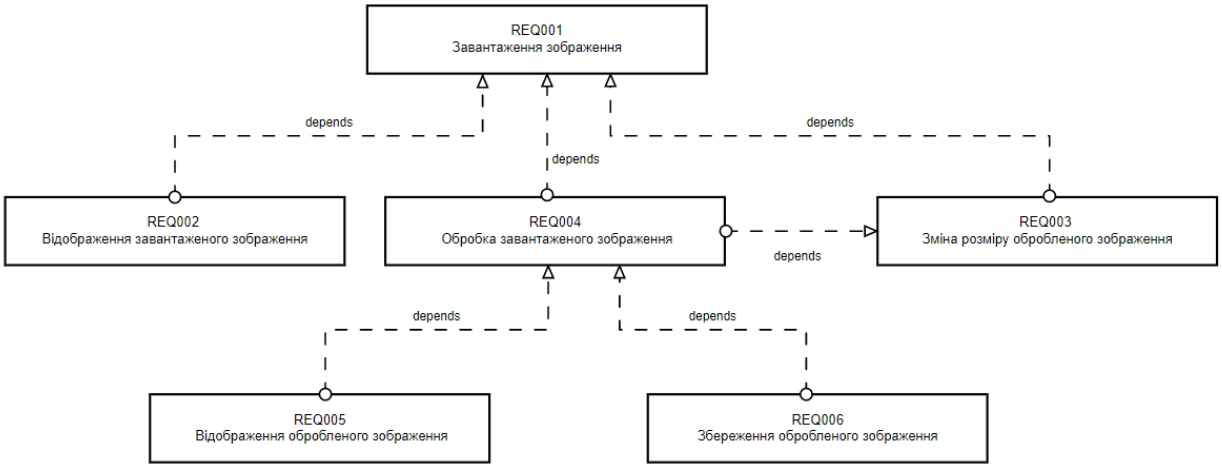


Рисунок 1.13 – Діаграма залежності між функціональними вимогами

Взаємозв'язки між вимогами розроблюваного програмного забезпечення зображені на рисунку 1.14.

	REQ001 Завантаження зображення	завантаженого зображення	REQ003 Зміна розміру обробленого зображення	REQ004 Обробка завантаженого зображення	REQ005 Відображення обробленого зображення	REQ006 Збереження обробленого зображення
REQ001 Завантаження зображення						
REQ002 Відображення завантаженого	■					
REQ003 Зміна розміру	■					
REQ004 Обробка	■		■			
REQ005 Відображення				■		
REQ006 Збереження				■		

Рисунок 1.14 – Матриця залежності між вимогами додатку

Взаємозв'язки між вимогами застосунку і варіантами використання відображено на рисунку 1.15.

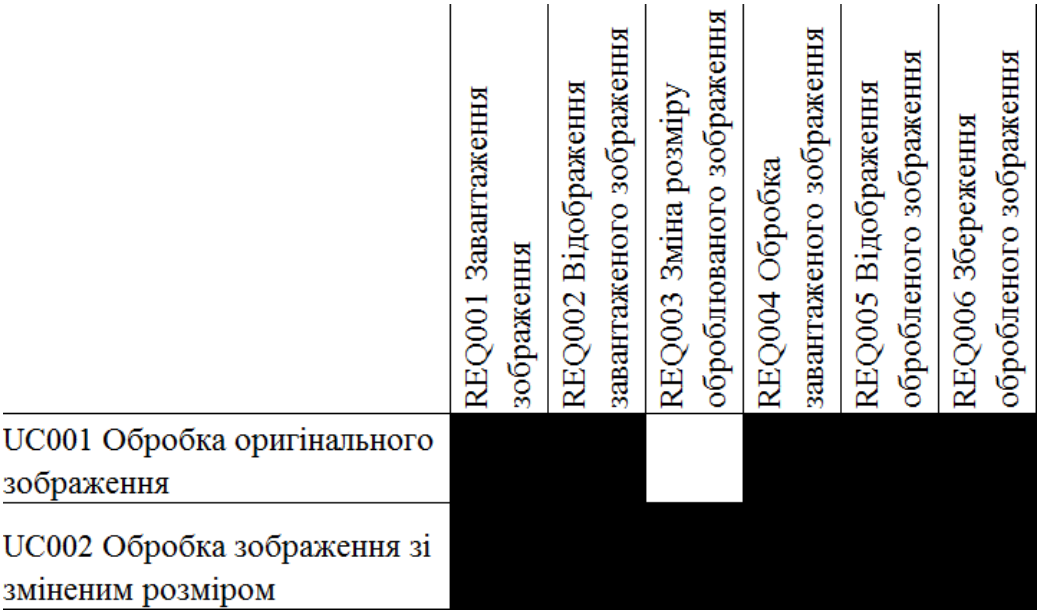


Рисунок 1.15 – Матриця залежності між вимогами застосунку і варіантами використання

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

- локалізація інтерфейсу – англійська;
- підтримувана версія браузера: Google Chrome 49 або вище;
- для передачі даних між клієнтом та серверами повинні використовуватись лише провірені та надійно захищені канали передачі данних.

1.4.3 Постановка комплексу завдань модулю

Розроблюване програмне забезпечення призначене для вирішення задачі деблюрінга зображень.

Мета створення даної роботи – створення алгоритму по деблюрінгу зображень з використанням генеративних змагальницьких мереж та його програмна реалізація в вигляді веб-додатка.

Для досягнення даної мети необхідно взяти за основу найкращий існуючий алгоритм деблюрінгу зображень, проаналізувати його слабкі сторони та найти технічні рішення для їх його покращення.

Розроблений застосунок повинен працювати на пристроях зі встановленою браузером Chrome.

1.5 Висновки по розділу

В першому розділі введено основні математичні моделі та розглянуті основі підходи до деблюрінга зображень. Оскільки непараметричні моделі використовують ітеративні алгоритми для оцінки розмиття зображення, то їх використання в реальних програмних додатках мало можливе. Натомість параметричні моделі здатні обробляти зображення з великою роздільною здатність в реальному часі при схожих або навіть кращих результатах порівняно з непараметричними моделями. Було описано алгоритм DeblurGAN який являються одним з найкращих представників параметричних моделей в термінах якості та швидкості.

Проаналізовано основні програмні реалізації алгоритмів деблюринг зображень та на їх основі були розроблені основні функціональні та не функціональні вимоги до розроблюваного програмного забезпечення. На їх основі було виконано постановку комплексного завдання.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

2.1.1 Алгоритм

У даній роботі я взяв алгоритм DeblurGAN за основу для генератора в алгоритмі CycleGAN, який покликаний обійти всі обмеження та недоліки DeblurGAN.

2.1.1.1 CycleGAN

Однією з істотних переваг для CycleGAN над іншими алгоритмами генеративних змагальницьких мереж є те, що він є не потребує попередньо підготовленого набору даних [28] з попартоною відповідність чіткого та розмитого зображення, а отже якість фінального рішення напряму не залежить від якості та складності побудованого алгоритму генерації розмиття зображення. Алгоритм CycleGAN складається з двох генераторів та двох дискримінаторів кожен з яких відповідає за перетворення даних з одного розподілу в інше та відрізнення згенерованих даних від реальних. Архітектура CycleGAN наведена на рисунку 2.1.

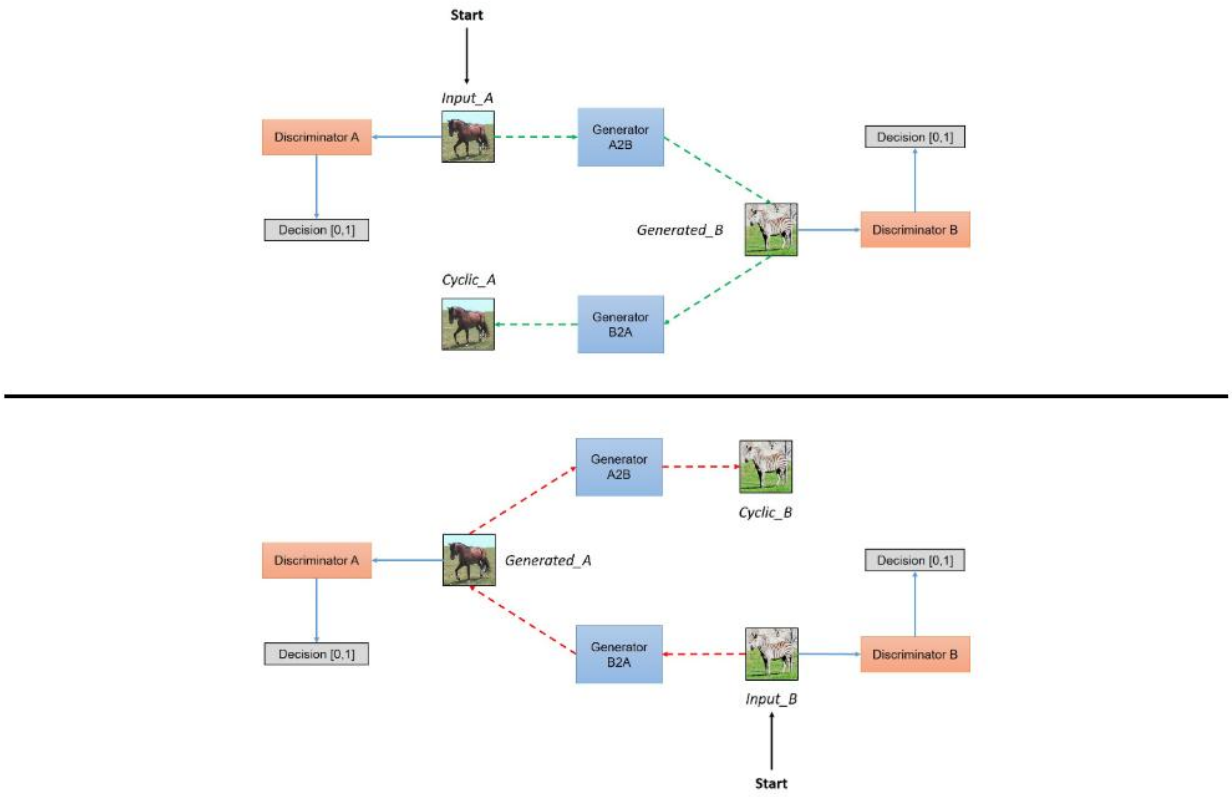


Рисунок 2.1 – Архітектура CycleGAN

2.1.1.2 Деталі реалізації

Фінальна архітектура підходу наведена на рисунку 2.2.

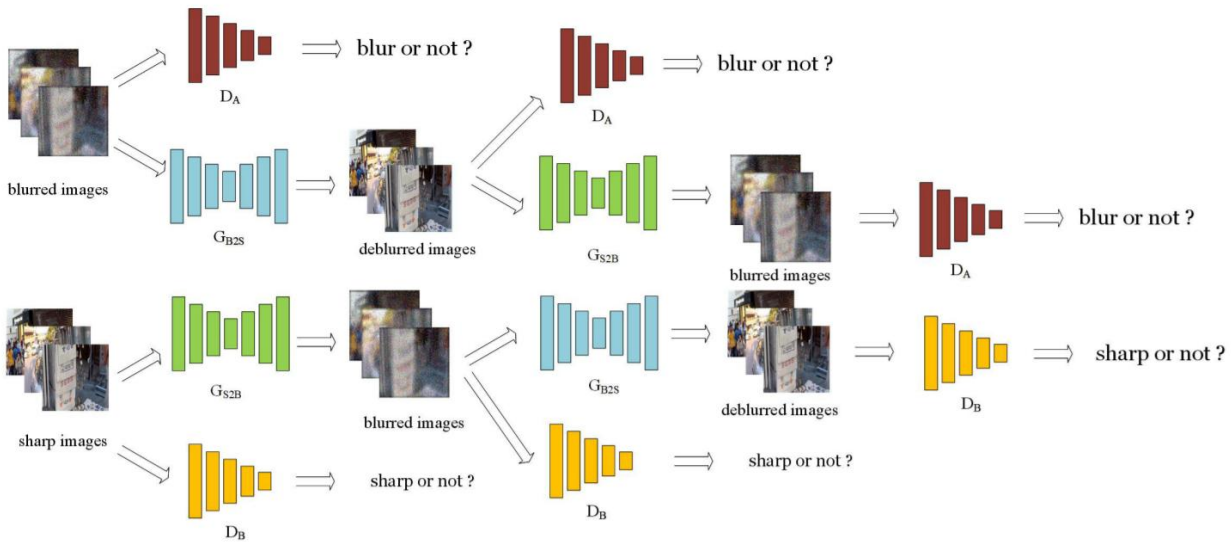


Рисунок 2.2 – Архітектура запропонованого підходу

Для її програмної реалізації я використав фреймворк TensorFlow [29]. Архітектура генераторів аналогічна до архітектури генератора DeblurGAN. На

вхід генератора подається кольорова картинка зі фіксованою розмірністю та на виході генераторів отримуємо також кольорову картинку з тієї ж розмірністю. Під час тестування вхідна розмірність фіксована та складає 1280 на 720 пікселів. Аналогічно до Style GAN [30] під час навчання вхідна розмірність поступово збільшується зі 128 на 72 пікселів до 1280 на 720 пікселів, що дозволяє скоротити час навчання в 25 раз. Аналогічно до DeblurGAN я використав преднавчену на ImageNet [31] мережу VGG-19 [32] в якості дискримінатора. Навчання відбувалося шляхом оптимізації функціоналу, що представлений в DeblurGAN, за допомогою SGDR [33] та стохастичного градієнтного спуску з моментумом. Коефіцієнт навчання рівний 2e-3 та коефіцієнт моментуму рівний 9e-1. Навчання відбувалося на орендованій машині p3.16xlarge в AWS який має в своєму розпорядженні 8 Nvidia Tesla V100 графічних процесорів з сумарним об'ємом відеопамяті в 128 Гб. Великий об'єм відеопамяті дозволив обробляти 16 картинок з розмірністю 1280 на 720 пікселів паралельно. Для більш ефективнішого використання даних була реалізована підтримка розподіленого підрахунку статистик для слоїв пачкової нормалізації та нормалізації семплів.

2.1.1.3 Протокол порівняння моделей

Аналогічно до інших робіт, що пов'язані з деблюрінгом зображень, я буду порівнювати якість свої підходів з попередніми алгоритмами за допомогою індексу структурної схожості зображень – SSIM [13] [16] [17] [18] [19] [34]. SSIM обраховується за формулою 2.1 та приймає значення від -1 до 1, де 1 – зображення ідентичні, а -1 – зображення зовсім не схожі один на одного. Оскільки SSIM підраховується для певного вікна розміру N на N (N = 15 для даної роботи), то SSIM для всього зображення шляхом усереднення SSMI для усіх можливих положень вікна N на N на зображені. Обрахунок SSIM для заданих вікон x та y наведено у формулі 2.1:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2.1)$$

де $c_1 = (0.01 * 255)^2$, $c_2 = (0.03 * 255)^2$, μ – математичне сподівання відповідного вікна, σ – середньоквадратичне відхилення відповідного вікна.

2.1.1.4 Експериментальні результати

Для порівняння отриманого підходу з попередніми алгоритмів буде використаний набор даних GoPRO [35] на якому були презентовані результати DeblurGAN. Результати наведені в таблиці 2.1.

Таблиця 2.1 – Результати алгоритмів деблюрінга на даних GoPRO

Алгоритм	Час обробки, с	SSIM
Blind Image Deblurring Using Dark Channel Prior, NVIDIA [36]	13.31	0.8070
Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring, Seoul National University [37]	4.33	0.7059
DeblurGAN, Ukrainian Catholic University [19]	0.85	0.8716
DeblurGAN+, NTUU “Kyiv Polytechnic Institute”	0.67	0.9003

З таблиці 2.1 можна замітити, що представлений алгоритм, який отримав назву DeblurGAN+, опереджає свого попередника як по якості так і по швидкості обробки одного зображення. Кращі результати в термінах якості отримані за рахунок новітнього підходу до генерації тренувальних пар. Кращі швидкісні показники отримані за рахунок використання оптимізацій часу виконання, які наявні в фреймворку Tensorflow [29], та відсутні у фреймворку PyTorch [38], який використовувався при оригінальній імплементації DeblurGAN.

2.1.2 Веб-додаток

Для створення якісного програмного забезпечення необхідно виконати його попереднє детальне моделювання, що дозволить створити деталізовані процеси, що відбуваються в розроблюваному програмному забезпеченні, та детальну архітектуру програмного забезпечення. Для проектування усіх внутрішніх процесів, що відбуваються в системі, було обрано BPMN як методологію створення діаграм.

Наведена схема містить узагальнені процеси, які проходить користувач під час використання розроблюваного застосунку. Загальні процеси, які проходить користувач, включають процес завантаження зображення на обробку, його відображення користувачу, можливу зміну розміру оброблюваного зображення, обробку зображення, виведення обробленого зображення та його завантаження користувачем на локальну систему.

Процес обробки завантаженого зображення розробленою інформаційною системо зображено за допомогою BPMN діаграми на рисунку 2.3:

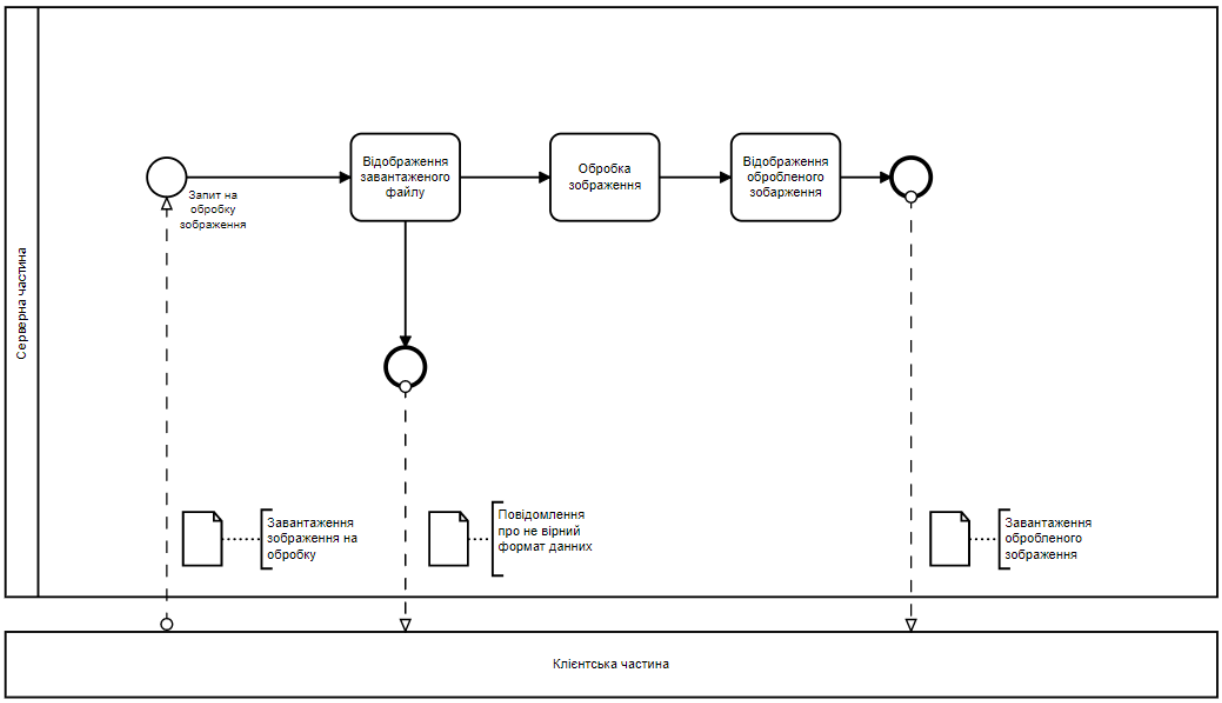


Рисунок 2.3 – Схема процесу обробки завантаженого зображення

Послідовний опис процесу обробки завантаженого зображення:

- клієнт надсилає зображення, яке необхідно обробити;
- сервер перевіряє завантажений формат завантаженого файлу;
- якщо завантажений формат не підтримується, то сервер надсилає відповідне повідомлення про помилку;
- сервер відображає завантажене зображення;
- сервер виконує обробку завантаженого зображення;
- сервер відображає оброблене зображення та посилає клієнту відповідне повідомлення.

2.2 Архітектура програмного забезпечення

Для вирішення поставленого завдання було обрано клієнт-серверну модель системи. Діаграма розгортання системи наведено у документах Діаграма розгортання програмного забезпечення. Схема структурна класів програмного забезпечення наведено у документі Схема структурна класів програмного забезпечення.

Взаємодія між компонентами системи здійснюється за допомогою REST-запитів, які натино підтримуються фрейморком Flask. REST являє собою шаблон між-мережевих протоколів доступ до інформаційних вузлів в мережі [45]. Він був створений та описаний Роєм Філдінгом у на початку двохтисячних років. В основі REST закладено усі центральні принципи функціонування Всесвітньої павутини, у тому числі і можливості HTTP 1.0. Оскікльки Філдінг займався розробкою REST-у паралельно з розробкою та затвердженням протоколу HTTP 1.1, то вони перейняли найкращі розробки їх попередника – протоколу HTTP 1.0. Ключовою ідеєю цих протоколі являється парадигма передачі даних, за якою усі данні повинні передаватися у вигляді невеликої кількості стандартизованих та уніфікованих форматів. Будь-який REST протокол не повинен підтримувати кешування та не повинен будь-як залежати від мережевого прошарку OSI моделі. Для досягнення даної мети протокол не

повинен зберігати ніякої інформації про стан. Даний підхід здатний забезпечити неймовірну горизонтальну масштабованість системи.

Яскравим представником антипатерну лдя REST являются підхіди, які основані на виклику не локальних процедур. Одним представником таких підходів являється підхід RPC, який дозволяє використовувати невелику кількість мережевих ресурсів для виконання великої кількості мережевих методів та являється дуже складним протоколом. При підході REST кількість методів і складність протоколу суворо обмежені, що призводить до того, що кількість окремих ресурсів має бути великою.

Для взаємодії з клієнтським застосуном використовують інтерфейс ImageProcessing, який описаний в таблиці 2.2.

Таблиця 2.2 – Методи інтерфейсу ImageProcessing

Назва метода	Аргументи	Значення, яке повертається	Коди помилок	Призначення методу
predict	image, processing_size	Image	0x0, 0x1	Обробка завантаженого користувачем зображення
load_model	path	-	0x2, 0x3	Завантаження ваг мережі з диску в память
dump_model	path	-	0x4, 0x5	Збереження ваг мережі на диск

Продовження таблиці 2.2

_preprocess	image, processing_size	Image	0x0, 0x1	Приватний метод. Пре-обробка вхідного зображення.
_predict	image	Image	0x5	Приватний метод. Обробка зображення мережею.
_postprocess	processed_image, origin_image	Image		Приватний метод. Зміна розміру обробленого зображення до розміру вхідного

Таблиця 2.3 – Методи інтерфейсу ImageProcessing

Код помилки	Системна назва помилки	Значення помилки
0x0	ERR_INVALID_IMAGE	Зображення являється не кольоровим
0x1	ERR_INVALID_SIZE	Зображення має малий розмір
0x2	ERR_FILE_NOT_FOUND	Вказаний файл не існує в файловій системі

Продовження таблиці 2.3

0x3	ERR_INVALID_FORMAT	Файл з вагами мережі має не вірний формат і не може бути завантажений
0x4	ERR_CAN_NOT_CREAT_FILE	Не можливо створити вихідний файл. Вказаний файл не існує в файловій системі або користувач не має прав на його створення
0x5	ERR_NOT_ENOUGH_SPACE	Не можливо створити файл з вагами, оскільки в файловій системі не достатньо місця для цього

Структури і типи даних, які відправляються і отримуються у запитах, вказано в таблиці 2.4.

Таблиця 2.4 – Структури і типи даних

Назва типу	Структура та/або тип даних	Опис даних
Image	np.ndarray	Структура містить зображення в вигляді тензора розмірності 3, кожен елемент даного тензору приймає значення в діапазоні від 0 до 255, а самий тензор має тип np.uint8.

ImageRes	{
	“width”: integer,
	“heihgt’: integer,
	}
		Дана структура даних призначення для передачі інформації про бажаний розмір оброблюваного зображення. Всі значення в дані структурі повинні мати позитивні цілі значення.

2.3 Конструювання програмного забезпечення

Програмне забезпечення було розроблене з використання наступних технологій та фрейморків:

- Python;
- Numpy;
- Tensorflow;
- Flask.

Давайте розглянемо їх переваги та недоліки для конструювання даного програмного забезпечення.

2.3.1 Python

Python був розроблений, щоб бути легким для розуміння і цікавим у використанні (його назва походить від Monty Python, тому багато його посібників для початківців посилаються на нього). Python набув популярності за те, що він є першокласною мовою, яка повністю замінила собою Java як найпопулярнішу вступну мову у вищих університетах США. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована. [39]

Переваги: простий та зрозумілий синтаксис, форматування коду – частина синтаксису, швидка розробка будь-яких додатків.

Недоліки: повільніше виконання програм в порівняні з компільованими аналогами.

2.3.2 Numpy

Numpy – це основна бібліотека для наукових обчислень у Python. Вона надає абстракцію багатовимірного масиву та високопродуктивні інструменти для роботи з цими масивами. Не менш важливим фактором являється і те що будь-який алгоритм який може бути виражений в основному як послідовність операцій над масивами і матрицям використанням Nympy працює настільки ж швидко як еквівалентний код написаний на C, але зі значно зрозумілішим синтаксисом на мові Python та меншим об’ємом написаного коду. [40]

Переваги: швидкість робити, простота маніпуляцій зі структурованими даними.

Недоліки: відсутні.

2.3.3 Tensorflow

TensorFlow — фреймворк для написання та налагодження алгоритмів глибинного машинного навчання. TensorFlow був розроблена компанією Google для задоволення внутрішніх потреб корпорації в системах, які базуються на побудові та навчанні глибоких нейромереж для роботи в реальному часі з надвеликими масивами інформації. TensorFlow наразі застосовують як для досліджень, так і для розробки продуктів Google, часто замінюючи на ролі його закритого попередника DistBelief. TensorFlow було розроблено командою Google Brain для внутрішнього використання в Google, поки не було вирішено випустити його під відкритою ліцензією Apache 2.0 9 листопада 2015 року.

TensorFlow дозволяє розробникам створювати ациклічні графи потоків тензорів – багатовимірних структур даних, що описують, як дані переміщуються через граф – певний ряд вузлів обробки. Кожен вузол у графі представляє математичну операцію кожен з яких містить з'єднання з декількома тензороми.

TensorFlow надає для користувачів мови Python зручне та зрозуміл API. Фактичною самі математичні операції не виконуються в Python. Бібліотеки TensorFlow, написані на високопродуктивні мові C++, а Python лише координує роботу між його частинами і надає абстракції програмування високого рівня, щоб поєднати їх разом.

Програми TensorFlow можна запускати на більшості платформах та будь-яких процесорах, які є зручними: локальна машина, кластер у хмарі, пристрої iOS і Android, центральні процесори або графічні процесори. Якщо ви користуєтеся персональною хмарою від Google, ви можете запустити TensorFlow на спеціальному процесорі TensorFlow (TPU) для отримання більшого неймовірного прискорення в обчисленнях. Отримані моделі, які створені в TensorFlow, можуть бути розгорнуті на більшості пристроїв, де вони можуть використовуватися для прогнозування. В червні 2016 року Джефф Дін з

Google заявив, що TensorFlow згадували 1500 репозиторіїв на GitHub, лише 5 з яких були від Google. [41]

Переваги: швидкість робити, простота в виконані маніпуляцій над тензорами, наявність пачкової обробки тензорів, підтримка розширень на мові C++.

Недоліки: складність налаштування програмних додатків при використанні складної архітектури.

2.3.4 Flask

Flask — фреймворк для мови програмування Python, який спрощує розробку веб-додатків та розповсюджується відповідно до умов ліцензії BSD.

Останнє стабільне оновлення для Flask було випущено в грудні 2016 та мало номер 0.12. Найбільш популярними проектами, які базуться на фрейморку Flask являються сайти Pinterest, LinkedIn, а також сторінка спільноти Flask.

Flask вважається міні-фрейморком, оскільки він не вимагає спеціальних сторонніх засобів чи бібліотек для розробки програмного забезпечення на даному фреймворку. У ньому присутні усі необхідні рівні абстракцій для роботи з базами даних, роботою з файлами та інші додатки, які надають необхідні функції. Також Flask має підтримку розширень, які забезпечують додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. Розширення оновлюються частіше аніж базовий код [42].

Ключовими властивостями Flask являються:

- простота в розробці та налагоджені програмного забезпечення;
- сумісність юніт-тестів з коробки;
- система керування запитами REST;
- підтримка патернів Jinja2;

- підтримка cookie на стороні веб-браузера;
- стандартизований по WSGI 1.0;
- функціональна підтримка багатьох мов;
- документованість;
- підтримка Google App Engine з коробки;
- наявність розширень від сторонніх розробників.

Переваги: швидкість робити, простота в використанні, RESTful API, докладна документація, підтримка Unicode.

Недоліки: складність в створенні складних багаторівневих веб-додатків та організація їх архітектури в термінах Flask.

2.4 Аналіз безпеки даних

Розроблюване програмне забезпечення підпадає під дію законопроекту GPDR [43] про захист персональної інформації громадян Європейського Союзу, оскільки більшість користувачі розроблюваного програмного забезпечення будуть завантажувати персональні фотографії у систему для їх обробки, тому необхідно приділити багато уваги для їх відповідно захисту.

У розроблюваному програмну забезпечення відсутня база даних і отже жодна інформація про користувачів або їхня історія активності не може бути викрадена, тому єдині спроби по викраденню користувацької інформації можуть мати місце лише під час передачі інформації від сервера до користувача і навпаки. Захист даних користувача гарантує протокол обміну REST, який працює поверх HTTPS та вважається захищеним протоколом обміну інформацією.

Для подальшого підвищення захисту персональної інформації та унеможливлення будь-яких спроб по їх викраденню можлива міграція з Tensorflow [29] на Tensorflow.js [44], який являється JavaScript розширенням Tensorflow [29], який здатний виконуватися в браузері на стороні клієнта і отже відпадає необхідність в передачі даних користувача через мережу Інтернет.

Проте проект Tensorflow.js [44] являється доволі молодим, то у ньому відсутні усі можливості фреймворку Tensorflow [29], а отже і відсутня підтримка всіх операції, які можна виконати на тензорами. Власне відсутність ряду ключових операції, які наявні в DeblurGAN+, унеможливлюють реалізація даного програмного продукту на Tensorflow.js [44], проте дана ситуація може змінитися уже в найближчі 1-2 роки, а отже існує велике вікно для подальших покращень безпеки даних для даної роботи.

2.5 Висновки по розділу

У даному розділі було проаналізовано бізнес-процеси інформаційної системи та проілюстровано з використанням діаграм BPMN. Було розроблено принципову схему клієнт-серверної моделі, описано і розроблено API та описано переваги та недоліки обраного набору засобів розробки. Проаналізовано безпеку користувацьких даних в інформаційні системі.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування розробленого функціоналу програмного забезпечення являється невід'ємною часткою циклу його розробки в усіх методологія програмування, оскільки воно здатне гарантував чітку відповідність до технічного завдання, коректність реалізації усіх ключових підсистем, наявність всіх заявлених можливостей та відсутність будь-яких помилок у розроблюваному програмному забезпеченні.

Всі підходи до тестування можуть бути поділено на ручне або автоматичне. Автоматизація тестування дуже спрощує весь цикл тестування, розробки програмного забезпечення та зменшує час між вивільненням інкрементальних змін для тестування, так як автоматичні тести гарантують, що зміни в уже готовому та протестованому коді програмного продукту не призвели до невірного функціонування вже розробленого та протестованого функціоналу, що дозволяє впевнено та швидко вносити зміни в швидкоростучу кодову базу.

При розробці алгоритму по деблюрігу зображень, тестування є надзвичайно важливим, оскільки розроблений алгоритм являється комплексним програмним продуктом та вимагає перевірки усіх часткових випадків. Проте, оскільки розроблений алгоритм являється ймовірнісним, то не можливо гарантувати 100% не відмовну його роботу. Проте навіть при таких жорстких обмеження на можливі набори тестів, які можуть бути використанні при тестуванні програмного додатку, його тестування все ж таки не є неможливим. Перш за все існує можливість створення елементарних одиничних тестів, які будуть перевіряти лише певні компоненти програмного забезпечення.

Ключовими типами тестування даного програмного продукту є навантажувальне тестування. Це обумовлено високою обчислювальною складністю розробленого алгоритму деблюрінга.

Навантажувальне тестування дозволяє перевірити швидкодію усіх компонентів розроблюваного програмного забезпечення, що можуть бути чутливими до навантаження на систему.

Також будуть протестовані наступні компоненти розробленого програмного забезпечення:

- тестування часу відповіді на запити;
- тестування API сервера по обробці зображень;
- тестування алгоритму обробки зображень;
- тестування функціоналу збереження результату у файловій системі;
- тестування функціоналу по попередньому зміні розміру картинки до заданого користувачем (зі збереженням співвідношення сторін);
- тестування роботи інтерфейсу користувача;
- навантажувальне тестування сервера по обробці зображень.

Тестування цих функцій дозволить повністю перевірити отриманий програмний продукт на відповідність функціональним вимогам.

3.2 Опис процесів тестування

Тестування відбуватиметься у режимі чорної коробки. Так як, у даному продукті дуже важлива його невідновна робота для різних наборі вхідних даних, тому необхідно щоб розроблений функціонал був протестовании як при в позитивних, так і в негативних умовах. Так як велику частину функціоналу продукту складно перевірити через ймовірнісну поведінку алгоритму та недоцільність його ручного тестування через необхідність перевірки великої кількості прикладів.

Будуть проводитись наступні типи тестів:

- димне тестування роботи API алгоритму;

- димне тестування роботи API сайту;
- інтеграціи не тестування компонентів програмного додатку;
- навантажувальне тестування серверу обробки зображень.

Навантажувальне тестування найкритичнішим компонентом для даного продукту, тому и ого важливо проводити під час як всіх етапі розробки програмного забезпечення, оскільки існують чіткі вимоги до швидкодії розроблювального програмного продукту. Для проведення навантажувального буду проведено генерація великого набору даних та апроксимація пікової та медіанної швидкості обробки на даному наборі даних. Даний підхід використовується по тій причині, що найбільш популярніший програмний додаток для проведення навантажувального тестування JMeter не підходить для проведення навантаження з великою кількістю різноманітних зображень, які необхідно відправляти через мережу.

3.3 Опис контрольного прикладу

Повний опис процесу тестування та наявних сценаріїв для тестування вказано в додатку В “Програма та методика тестування”, а в даному розділі розглянуто проведення тестування на прикладі сценарію перевірки коректності деблюрінгу зображення. Даний приклад наведено у таблиці 3.1.

Таблиця 3.1 – Опис контрольного прикладу

Мета тесту	Перевірка коректності деблюрінга зображення з явними елементами розмитості вхідного
Початковий стан	Система готова до обробки вхідних зображень. Зображення bear.jpeg завантажено в файлову систему
Вхідні дані	Зображення bear.jpeg

Продовження таблиці 3.1

Схема проведення тесту	Запустити скрипт по обробці зображення та обрати файл bear.jpeg як вхідне зображення та вказати файл beaver_clean.jpeg як вихідне.
Очікуваний результат	В файловій системі з’являється файл bear_clean.jpeg, який містить контент аналогічний до контенту зображення bear.jpeg, проте без елементів розмитості.
Кінцевий результат	Після проведення даного тесту стан системи не повинен змінитись

Розглянутий тест відноситься до тестів базової функціональності та виконується без графічного інтерфейсу користувача в командній стрічці. Для його виконання слід запустити відповідний скрипт тестування алгоритму, обрати зображення в файловому провіднику системи та задати шлях для збереження результату. Після цього слід перевірити чи співпадає отриманий результат з очікуваним результатом. Результат виконання тесту наведеного в таблиці 3.1 зображено на рисунках 3.1 та 3.2.

Рисунок 3.1 – Вхідне зображення до контрольного прикладу

Рисунок 3.2 – Результуюче зображення до контрольного прикладу

3.4 Висновок до розділу

В даному розділі було описано підходи та процеси тестування розробленого програмного забезпечення. Описано основні компоненти, які мають бути протестовані для впевнення, що фінальний продукт відповідає усім вимогам якості.

Описано в розгорнутому вигляді контрольний приклад для перевірки коректності роботи розробленого програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення необхідно виконати наступні кроки:

1) Інсталювати Python 3.6.5 або вище в системі;
2) Інсталювати залежності виконавши команду «pip install -r requirements.txt»;
3) Запустити сервер по обробці зображень виконавши команду «python3 app.py»;
4) Інсталювати node.js та npm;
5) Інсталювати залежності виконавши команду “nmp install dependencies”;
6) Виконати команду “nmp start” для того щоб стартувати веб-сервер.

4.2 Робота з програмним забезпеченням

Розроблене програмне забезпечення передбачає наступний сценарій використання:

1) Відкриття додатку в браузері;
2) Завантаження зображення на обробку програмним забезпеченням;
3) Обрання розміру оброблюваного зображення;
4) Натискання кнопки «Обробити»;
5) Завантаження обробленого зображення.

4.3 Висновок до розділу

У даному розділі продемонстровано схему розгортання системи та описано сценарій використання розробленої інформаційної системи.

ВИСНОВКИ

У ході дипломного проекту було проведено аналіз проблематики деблюрінг зображень: описано математичні моделі, проаналізовано існуючі підходи, описано їхні переваги та недоліки.

Було спроектовано і розроблено новий підхід до генерації синтетичних зображень, що дозволило покращати уже існуючий алгоритм DeblurGAN. На основі розробленого алгоритму було розроблено веб-додаток для візуального представлення розробленого алгоритму.

Також було проведено аналіз якості інформаційної системи, проведено тестування з подальшим усуненням усіх знайдені недоліків.

Розроблено необхідну проектну документацію, схеми процесів застосунку, варіантів використань та керівництво для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1) “Machine learning attachs against Asirra Captcha”, [Електронний ресурс]. – 2008. – Режим доступу: https://crypto.stanford.edu/~pgolle/papers/dogcat.pdf

2) Dogs versus cats Kaggle competition on Asirra dataset [Електронний ресурс]. – 2019. – Режим доступу: https://www.kaggle.com/c/dogs-vs-cats

3) Khmag, A., Al Haddad, S., Ramlee, R., Kamarudin, N., Malallah, F.L.: Natural image noise removal using nonlocal means and hidden markov models in transform domain. The Visual Computer 34(12), 1661–1675 (2018)

4) Fan, Q., Shen, X., Hu, Y.: Detail-preserved real-time hand motion regression from depth. The Visual Computer pp. 1–10 (2018)

5) Cambra, A.B., Murillo, A.C., Munoz, A.: A generic tool for interactive complex image editing. The Visual Computer 34(11), 1493–1505 (2018)

6) Guan, H., Cheng, B.: How do deep convolutional features affect tracking performance: an experimental study. The Visual Computer 34(12), 1701–1711 (2018)

7) Yu, Z., Liu, Q., Liu, G.: Deeper cascaded peak-piloted network for weak expression recognition. The Visual Computer 34(12), 1691–1699 (2018)

8) Ineichen, P.: High turbidity solis clear sky model: Development and validation. Remote Sensing 10(3), 435 (2018)

9) Sun, Z., Zhang, Q., Li, Y., Tan, Y.a.: Dppdl: a dynamic partialparallel data layout for green video surveillance storage. IEEE Transactions on Circuits and Systems for Video Technology 28(1), 193–205 (2018)

10) Hobbs, J.B., Goldstein, N., Lind, K.E., Elder, D., Dodd III, G.D., Borgstede, J.P.: Physician knowledge of radiation exposure and risk in medical imaging. Journal of the American College of Radiology 15(1), 34–43 (2018)

11) Ian Goodfellow, Yoshua Bengio, Aaron Courville. “Deep learing”, 2016

12) “Understanding Deep Image Representations by Inverting Them”, 2015 [Електронний ресурс]. – 2019. – Режим доступу: https://www.robots.ox.ac.uk/~vedaldi/assets/pubs/mahendran15understanding.pdf

13) Liu, G., Chang, S., Ma, Y.: Blind image deblurring using spectral properties of convolution operators. IEEE Transactions on image processing 23(12), 5047–5056 (2014)

14) Chandramouli, P., Jin, M., Perrone, D., Favaro, P.: Plenoptic image motion deblurring. IEEE Transactions on Image Processing 27(4), 1723–1734 (2018)

15) Kotera, J., Sroubek, F.: Motion estimation and deblurring of fast moving objects. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 2860–2864. IEEE (2018)

16) Krishnan, D., Fergus, R.: Fast image deconvolution using hyperlaplacian priors. In: Advances in Neural Information Processing Systems, pp. 1033–1041 (2009)

17) Wang, R., Tao, D.: Training very deep cnns for general non-blind deconvolution. IEEE Transactions on Image Processing 27(6), 2897–2910 (2018)

18) Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep cnn denoiser prior for image restoration. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2 (2017)

19) Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: Blind motion deblurring using conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2261–2269 (2017)

20) Simonyan, K. and A. Zisserman (2014). “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: ArXiv e-prints. arXiv: 1409.1556 [cs.CV]

21) Ronneberger, O., P. Fischer, and T. Brox (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: ArXiv e-prints. arXiv: 1505.04597 [cs.CV]

22) Sun, Jian et al. “Learning a Convolutional Neural Network for Nonuniform Motion Blur Removal”. In: Szeliski, Richard (2010). Computer Vision: Algorithms and Applications. 1st. New York, NY, USA: Springer-Verlag New York, Inc. ISBN: 1848829345, 9781848829343

23) Chakrabarti, Ayan (2016). “A neural approach to blind motion deblurring”. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). ISBN: 9783319464862.

24) Boracchi, G. and A. Foi (2012). “Modeling the Performance of Image Restoration From Motion Blur”. In: Image Processing, IEEE Transactions on 21.8, pp. 3502 –3517. ISSN: 1057-7149. DOI: 10.1109/TIP.2012.2192126.

25) Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Joshua (2014). «Generative Adversarial Networks». arXiv:1406.2661 [cs.LG].

26) Генеративна змагальна мережа. [Електронний ресурс]. – 2019. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D0%B7%D0%BC%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0

27) SmartDeblur. [Електронний ресурс]. – 2019. – Режим доступу: http://smartdeblur.net/

28) Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv:1703.10593 [cs.CV]

29) Tensorflow. [Електронний ресурс]. – 2019. – Режим доступу: https://www.tensorflow.org/

30) Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv: 1812.04948 [cs.NE].

31) Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575 [cs.CV]

32) Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV]

33) Ilya Loshchilov, Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983 [cs.ML]

34) SSIM. [Електронний ресурс]. – 2019. – Режим доступу: https://en.wikipedia.org/wiki/Structural_similarity

35) GoPRO dataset. [Електронний ресурс]. – 2019. – Режим доступу: https://drive.google.com/file/d/1H0PIXvJH4c40pk7ou6nAwoxuR4Qh_Sa2/view?usp=sharing

36) Jinshan Pan, Deqing Sun, Hanspeter Pfister, Ming-Hsuan Yang. Blind Image Deblurring Using Dark Channel Prior. CVPR 2016.

37) Seungjun Nah, Tae Hyun Kim, Kyoung Mu Lee. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. CVPR 2017.

38) PyTorch. [Електронний ресурс]. – 2019. – Режим доступу: https://pytorch.org/

39) Python (programming language) [Електронний ресурс]. – 2019. – Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language)

40) NumPy [Електронний ресурс]. – 2019. – Режим доступу: https://en.wikipedia.org/wiki/NumPy

41) Tensorflow [Електронний ресурс]. – 2019. – Режим доступу: https://uk.wikipedia.org/wiki/TensorFlow

42) Flask [Електронний ресурс]. – 2019. – Режим доступу: https://uk.wikipedia.org/wiki/Flask

43) GDPR [Електронний ресурс]. – 2019. – Режим доступу: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

44) Tensorflow.js [Електронний ресурс]. – 2019. – Режим доступу: https://www.tensorflow.org/js

45) REST [Електронний ресурс]. – 2019. – Режим доступу: https://uk.wikipedia.org/wiki/REST

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

«Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем»

Опис програми

КПІ.ІП-5106.045490.02.13

“ПОГОДЖЕНО”

Керівник проекту:

_____ Г.В. Ісаченко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ О.А. Зарічковий

Київ – 2019 року

Тексти програмного коду

Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем

(Найменування програми (документа))

DVD-R

(Вид носія даних)

14 арк, 337101 Кб

(Обсяг програми (документа) , арк.,)

Київ – 2019

```
import tensorflow as tf
import cv2
import numpy as np

from typing import Tuple

from PIL import Image


def load_image(image_iobuffer, shape=None, max_size=None):
    image = Image.open(image_iobuffer)

    if max_size is not None:
        # Calculate the appropriate rescale-factor for
        # ensuring a max height and width, while keeping
        # the proportion between them.
        factor = float(max_size) / np.max(image.size)

        # Scale the image's height and width.
        size = np.array(image.size) * factor

        # The size is now floating-point because it was scaled.
        # But PIL requires the size to be integers.
        size = size.astype(int)

        # Resize the image.
        image = image.resize(size, Image.LANCZOS) # PIL.Image.LANCZOS is
one of resampling filter
```

```
        if shape is not None:
            image = image.resize(shape, Image.LANCZOS) # PIL.Image.LANCZOS
is one of resampling filter

        # Convert to numpy floating-point array.
        image = np.expand_dims(image, axis=0)
        image = image.astype(dtype=np.float32)
        return image



    def save_output_image(image_obj, output_image):
        # Ensure the pixel-values are between 0 and 255.
        image = np.clip(output_image, 0.0, 255.0)

        # Convert to bytes.
        image = image.astype(np.uint8)

        # # Write the image-file in jpeg-format.
        # with open(filename, 'wb') as file:
        #     PIL.Image.fromarray(image).save(file, 'jpeg')



    class TFSessionHandler(object):
        """
        TensorFlow Session handler.
        """
        def __init__(self, scope: str = ""):
            """
            :param scope: a name scope of model.
```

```
        """
        self.sess = tf.Session()
        self.scope = scope

    def __del__(self):
        self.sess.close()

    def _load_protobuf(self, protobuf_path: str) -> tf.Graph:
        """
        Load model from protobuf into graph.
        :param protobuf_path: path to model protobuf
        :return: graph with loaded model.
        """

        tf_graph = tf.Graph()

        with tf_graph.as_default():
            tf_graph_def = tf.GraphDef()
            with tf.gfile.GFile(protobuf_path, 'rb') as protobuf_file:
                serialized_graph = protobuf_file.read()
                tf_graph_def.ParseFromString(serialized_graph)
                tf.import_graph_def(tf_graph_def, name=self.scope)

        tf.contrib.graph_editor.copy(tf_graph, self.sess.graph)

        return self.sess.graph


class StyleTransferDemo(TFSessionHandler):
```

```
    """
    Style Transfer demo helper.
    """

    def __init__(self,
                 model_path: str,
                 input_shape: Tuple[int, int],
                 scope: str = ''):
        """
        :param model_path: path to model protobuf file.
        :param input_shape: model input shape (height, width).
        :param scope: a name scope of model.
        """

        super(StyleTransferDemo, self).__init__(scope=scope)
        self.model_path = model_path
        self.input_shape = input_shape
        self.model = self._load_model()
        self.scope = scope + '/' if scope else ''

        self.input_image_tensor = self.model.get_tensor_by_name(
            self.scope + 'content:0'
        )
        self.output_image_tensor = self.model.get_tensor_by_name(
            self.scope + 'styled_image:0'
        )

    def _load_model(self) -> tf.Graph:
        """
```

```
        Load model from serialized file and update session graph
        :return: graph with loaded model.
        """

        graph = self._load_protobuf(self.model_path)
        return graph

    def transform(self, frames_rgb: np.ndarray) -> np.ndarray:
        """
        Apply stylization batch of images.
        :param frames_rgb: batch of images.
        :return: styled images.
        """

        if frames_rgb.ndim == 3:
            frames_rgb = np.expand_dims(frames_rgb, axis=0)
        elif frames_rgb.ndim != 4:
            raise ValueError('Inputs must be 3 or 4 dimensional.'
                             'Get input with {} dims.'.format(frames_rgb.ndim))

        frames_rgb = [
            cv2.resize(frame,
                       dsize=(self.input_shape[1], self.input_shape[0]))
            for frame in frames_rgb
        ]

        styled_images = self.sess.run(
            fetches=self.output_image_tensor,
            feed_dict={
```

```
                self.input_image_tensor: frames_rgb
            })

        styled_images = np.clip(styled_images, a_min=0., a_max=255.)
        styled_images = styled_images.astype(dtype=np.uint8)
        styled_images = np.squeeze(styled_images)

        return styled_images

    def __call__(self, frames_rgb: np.ndarray) -> np.ndarray:
        return self.transform(frames_rgb)

import os
from io import BytesIO

from PIL import Image
from django.shortcuts import render, get_object_or_404
from rest_framework import status
from rest_framework.generics import GenericAPIView
from rest_framework.response import Response
from django.core.files.base import ContentFile

from .serializers import ImageUploadSerializer
from .utils import *
from .models import Image as ImageModel


class UploadProcessImageAPIView(GenericAPIView):
```

```
        def post(self, request, *args, **kwargs):
            serializer = ImageUploadSerializer(data=request.data)
            serializer.is_valid(raise_exception=True)
            data = serializer.validated_data
            input_image_file = data.get('original_file')

            input_image = load_image(input_image_file.file)
            style_slug = data.get('style')

            model_path = os.path.join(os.path.dirname(__file__),
'tf_models/{}.pb'.format(style_slug))
            transformer = StyleTransferDemo(model_path=model_path,
input_shape=(360, 640), scope='la_muse')

            output_image = transformer.transform(input_image)
            output_image = np.clip(output_image, 0.0, 255.0)
            output_image = output_image.astype(np.uint8)

            with BytesIO() as output:
                Image.fromarray(output_image).save(output, 'jpeg')
                output_image_data = output.getvalue()

            with BytesIO() as output:
                Image.open(input_image_file.file).save(output, 'jpeg')
                input_image_data = output.getvalue()

            image = ImageModel()
            new_filename = '{}_'.format(style_slug) + input_image_file.name
```

```
            image.original_file.save(input_image_file.name,
ContentFile(input_image_data), save=False)
            image.edited_file.save(new_filename,     ContentFile(output_image_data),
save=False)
            image.style = style_slug
            image.save()


            return                Response({'url':                image.edited_file.url},
status=status.HTTP_200_OK)


        import React, { Component } from 'react'
        import Background from './assets/seattle.jpg'
        import Try from './Try'
        import Output from './Output'

        import Spinner from './../Spinner'

        import LaMuse from './assets/la_muse.jpg'
        import RainPrincess from './assets/rain_princess.jpg'
        import TheScream from './assets/the_scream.jpg'
        import Udnie from './assets/udnie.jpg'
        import Wave from './assets/wave.jpg'

        import './index.css'

        const stylesConfig = [
          {
            name: 'La Muse',
            img: LaMuse,
            value: 'la_muse',
          },
          {
            name: 'Rain Princess',
            img: RainPrincess,
            value: 'rain_princess',
          },
          {
            name: 'The Scream',
            img: TheScream,
```

```
    value: 'scream',
  },
  {
    name: 'Udnie',
    img: Udnie,
    value: 'udnie',
  },
  {
    name: 'Wave',
    img: Wave,
    value: 'wave',
  },
]

const url = 'http://127.0.0.1:8000/'

class Main extends Component {
  constructor(props) {
    super(props)
    this.state = {
      loading: false,
      activeStyle: stylesConfig[0].value,
      responseUrl: '',
    }

    this.startLoading = this._startLoading.bind(this)
    this.finishLoading = this._finishLoading.bind(this)
    this.changeStyle = this._changeStyle.bind(this)
    this.load = this._load.bind(this)
  }

  _startLoading() {
    const { loading } = this.state
    if (!loading) {
      this.setState({
        loading: true,
      })
    }
  }

  _finishLoading() {
    const { loading } = this.state
    if (loading) {
      this.setState({
```

```
      loading: false,
    })
  }
}

_changeStyle(style) {
  this.setState({
    activeStyle: style,
  }, () => {
    this.load()
  })
}

_load(newFile) {
  const { activeStyle } = this.state

  this.startLoading()
  let file = newFile

  if (newFile) {
    this.setState({
      file: newFile,
    })
  } else {
    file = this.state.file
  }

  const header = new Headers({
    'Access-Control-Allow-Origin': '*',
    'Content-Type': 'multipart/form-data',
    'Access-Control-Allow-Credentials': 'true',
  })

  const formData = new FormData()
  formData.append('original_file', file, file.name)
  formData.append('style', activeStyle)

  const sentData = {
    method: 'POST',
    mode: 'cors',
    header: header,
    body: formData || '',
  }
```

```
        fetch(url, sentData)
          .then((response) => {
            return response.json()
          })
          .then((json) => {
            console.log('json', json)
            const { url } = json
            this.setState({
              responseUrl: url.slice(1),
            })
            this.finishLoading()

            const scrollTo = (scrollPosition, scrollDuration) => {
              let scrollStep = Math.abs(scrollPosition - window.scrollY) /
(scrollDuration / 15)
              const direction = scrollPosition > window.scrollY ? 1 : -1
              scrollStep *= direction
              const scrollInterval = setInterval(() => {
                if (window.scrollY !== scrollPosition) {
                  window.scrollBy(0, scrollStep)
                } else {
                  clearInterval(scrollInterval)
                }
              }, 15)
              setTimeout(() => {
                clearInterval(scrollInterval)
              }, scrollDuration * 1.1)
            }

            scrollTo(window.innerHeight, 500)
          })
          .catch((err) => {
            console.log('err', err)
            this.finishLoading()
            alert('Error', err)
          })
      }

      render() {
        const {
          loading,
          activeStyle,
          responseUrl,
        } = this.state
```

```
    const spinner = loading
      ? <Spinner />
      : null

    const style = {
      backgroundImage: `url(${Background})`,
    }

    return (
      <div>
        {spinner}
        <div
          className="main"
          style={style}
        >
          <div className="info">
            <div className="title">
              Deep Style
            </div>
            <div className="desc">
              Using deep learning for artistic style transfer
            </div>
          </div>
          <Try
            load={this.load}
          />
        </div>
        <Output
          stylesConfig={stylesConfig}
          activeStyle={activeStyle}
          changeStyle={this.changeStyle}
          responseUrl={`${url}${responseUrl}`}
          isHide={!responseUrl}
        />
      </div>
    )
  }
}

export default Main
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

«Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем»

Технічне завдання

КПІ.ІП-5106.045490.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ Г.В. Ісаченко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ О.А. Зарічковий

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: «Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем».

Галузь застосування: систем обробки сигналів.

Наведене технічне завдання поширюється на розробку «Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем» КПІ.ІП-5106.045490, котра використовується для деблюрінгу зображень та призначена для використання як частина систем обробки сигналів, що працюють зі зображеннями та відео.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання як частина систем обробки сигналів, що працюють зі зображеннями та відео.

Метою розробки є створення алгоритму деблюрінгу зображень.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання деблюрінг зображень. Розробку виконати на платформі Linux. Також необхідно передбачити вибір оброблюваного зображення з файлової системи користувача та загрузку обробленого зображення з сервера.

4.2 Вимоги до надійності

Програмне забезпечення повинно передбачити контроль введення інформації та захист від некоректних дій користувача.

4.3 Умови експлуатації

Програмне забезпечення повинно відповідати умовам експлуатації згідно СанПін 2.2.2.542 – 96. Розроблюване програмне забезпечення функціонує в автоматичному режимі та не вимагає обслуговування.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах. Мінімальна конфігурація технічних засобів:

- центральний процесор: 2x Intel Xeon Gold 6154 (3.0 GHz, 18 cores);
- оперативна пам’ять: 384 GB DDR4-3200;
- графічний прискорювач: 8x NVIDIA Tesla V100 (5120 CU, 640 TU), 16 GB HBM2 VRAM;
- постійна пам’ять: 512 GB.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix. Вхідні дані повинні бути представлені в наступному форматі: кольорова картинка в форматі jpeg або png та з розмірністю більшою за 128 пікселів по кожній зі сторін. Результати повинні

бути представлені в наступному форматі: кольорова картинка в форматі jpeg та з розмірністю еквівалентною до розмірності вхідного зображення. Програмне забезпечення повинно бути розроблено на мові програмування Python з використанням фреймворків Tensorflow та Flask в IDE PyCharm.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

- Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6);
- Технічне завдання;
- Керівництво користувача;
- Керівництво системного програміста;
- Програма та методика тестування;
- Графічна частина повинна бути виконана на 3х листах формату А3, котрі включаються у якості додатків до пояснювальної записки: Структура генеративної мережі, Схема структурна класів програмного забезпечення, Діаграма розгортання програмного забезпечення.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк	Звітність
1	Вивчення літератури за тематикою проекту	20.04.19	
2	Розробка технічного завдання	23.04.19	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	25.08.19	Специфікації програмного забезпечення
4	Проектування структури програмного забезпечення, проектування компонентів	28.04.19	Схема структурна програмного забезпечення та специфікація компонентів
5	Програмна реалізація програмного забезпечення	15.05.19	Вихідні тексти програмного забезпечення
6	Тестування програмного забезпечення	19.05.19	Тести, результати тестування
7	Розробка матеріалів текстової частини проекту	25.05.19	Пояснювальна записка.
8	Розробка матеріалів графічної частини проекту	25.05.19	Графічний матеріал проекту
9	Оформлення технічної документації проекту	25.05.19	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

«Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем»

Програма та методика тестування

КПІ.ІП-5106.045490.04.51

“ПОГОДЖЕНО”
Керівник проекту:
_____ Г.В. Ісаченко

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____ О.А. Зарічковий

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Web-ресурс по деблюрінгу зображень, який являє собою web-сайт, створений за допомогою фрейморку Flask на мові програмування Python з використанням технології React Native App.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок web-ресурсу;
- функціональна працездатність алгоритму деблюрінгу зображень;
- наявність доступу до бази дистанційних курсів в системі EDU;
- відповідність форматів та протоколів пересилання даних з веб-сервером;
- забезпечення належного рівня безпеки даних;
- зручність роботи з web-сайтом;
- відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування відбуватиметься у режимі чорної скриньки. Для надійності функціонал буде протестований як при позитивних, так і при негативних умовах. Перед початком тестування нової версії програмного продукту необхідно обов’язково провести димне тестування API сервера по обробці зображень та димне тестування інтерфейсу користувача.

Тестування відбувається на рівні «ітеграціного тестування». Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (ба-зове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу;

– тестування API;

– функціональне тестування коректності алгоритму деблюрінг.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

«Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем»

Керівництво користувача

КПІ.ІП-5106.045490.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ Г.В. Ісаченко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ О.А. Зарічковий

Київ – 2019 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Вхід на сайт

Перед початком роботи з програмним продуктом користувач повинен завантажити стартову сторінку веб-додаток (Рисунок 1.1).

Рисунок 1.1 – Стартова сторінка

1.2 Завантаження зображення

Після завантаження стартової сторінку користувачеві необхідно нажати на кнопку «Choose images» та обрати необхідний для обробки файл в файловому провіднику (Рисунок 1.2).

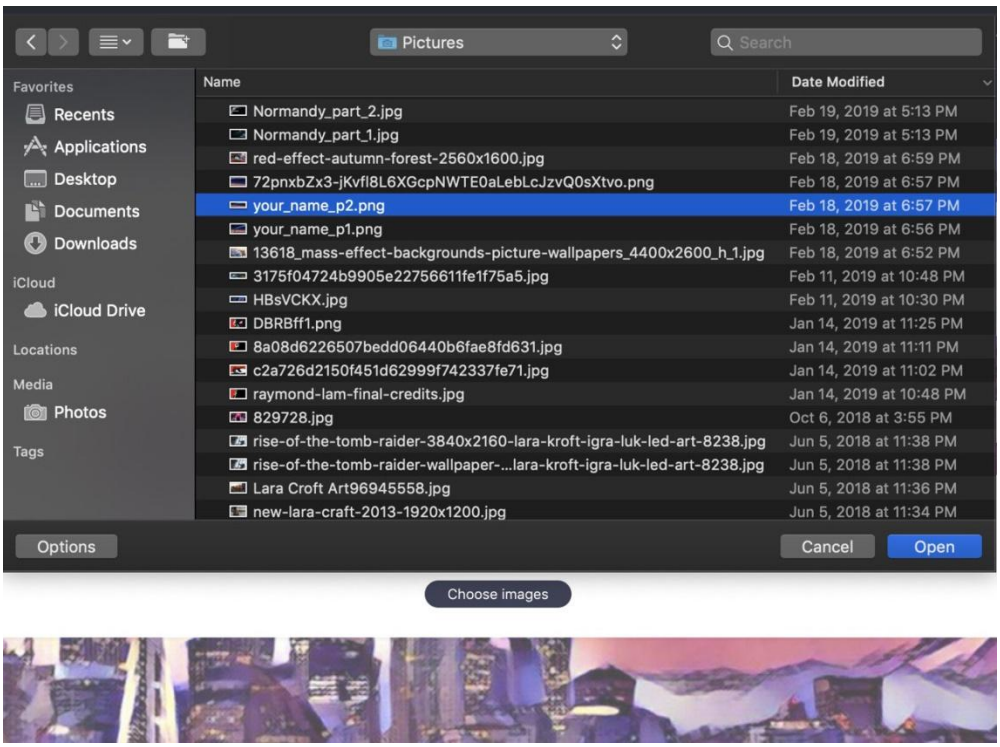
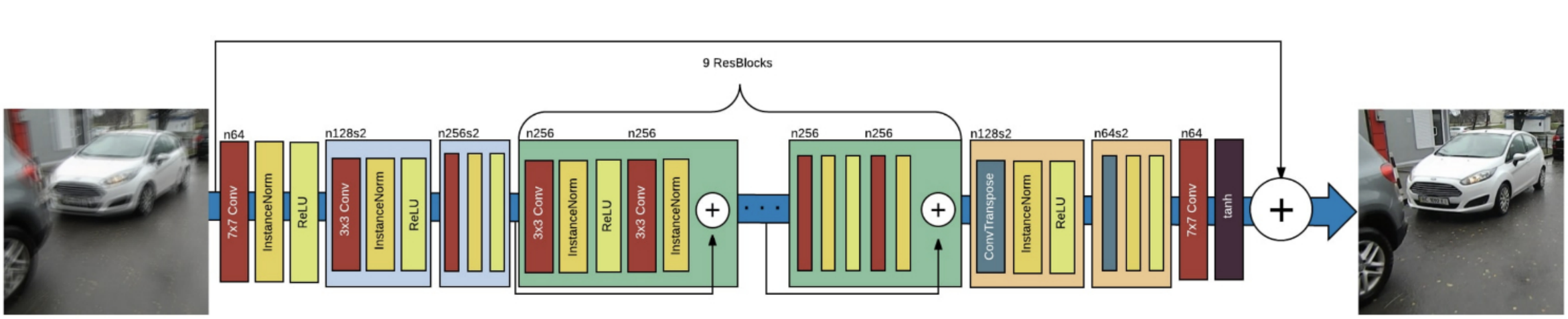


Рисунок 1.2 – Головна сторінка користувача

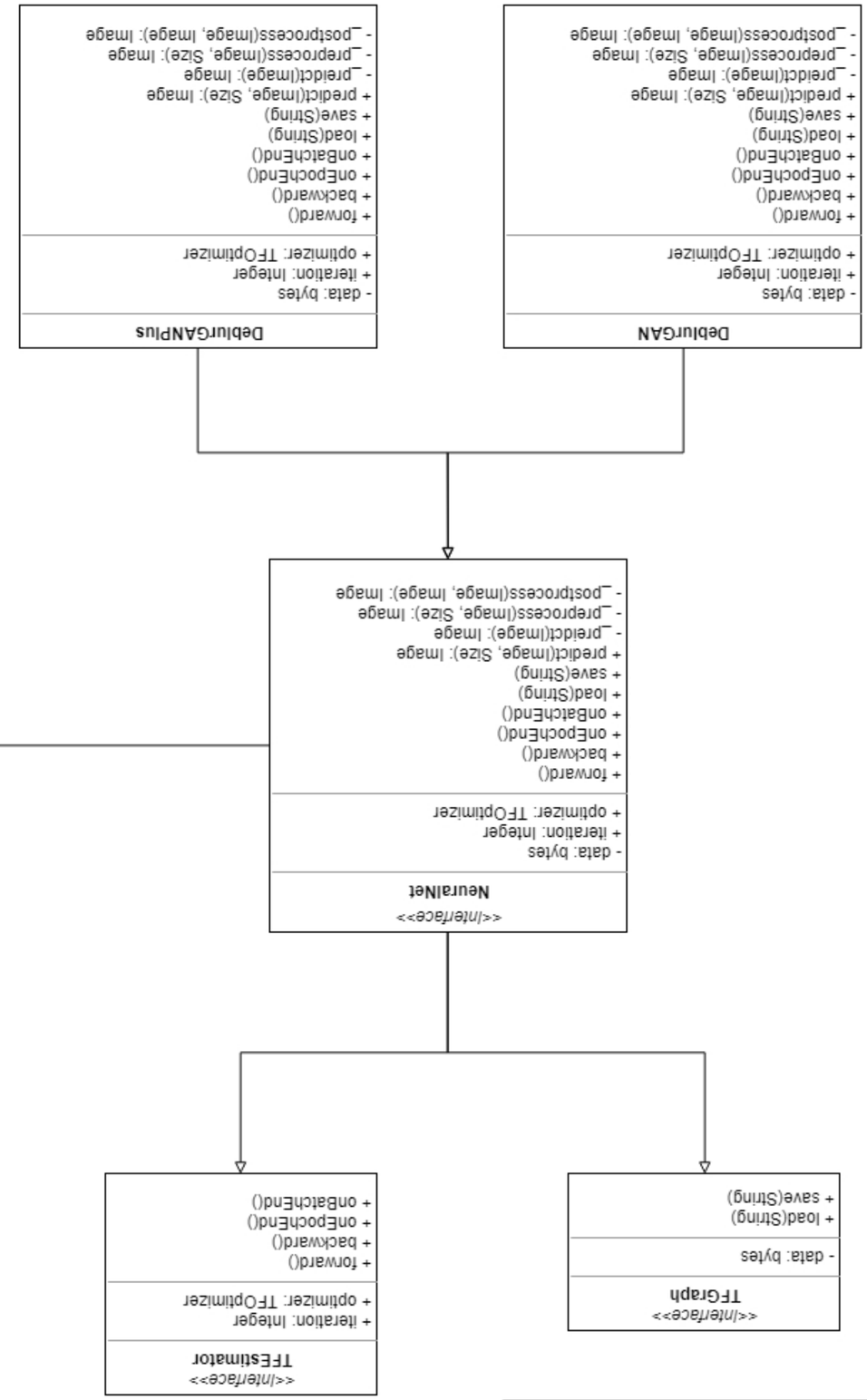
1.3 Обробка зображення

Після обрання зображення, воно буде передано на сервер для його обробки. Під час обробки зображення на екрані з’являється анімація, що свідчить про те що процес успішно розпочато, а через декілька секунд користувачеві буде відображено результат роботи програмного забезпечення (рисунок 1.3), а користувач може завантажити оброблене зображення.

Рисунок 1.3 – Результат роботи програмного додатку

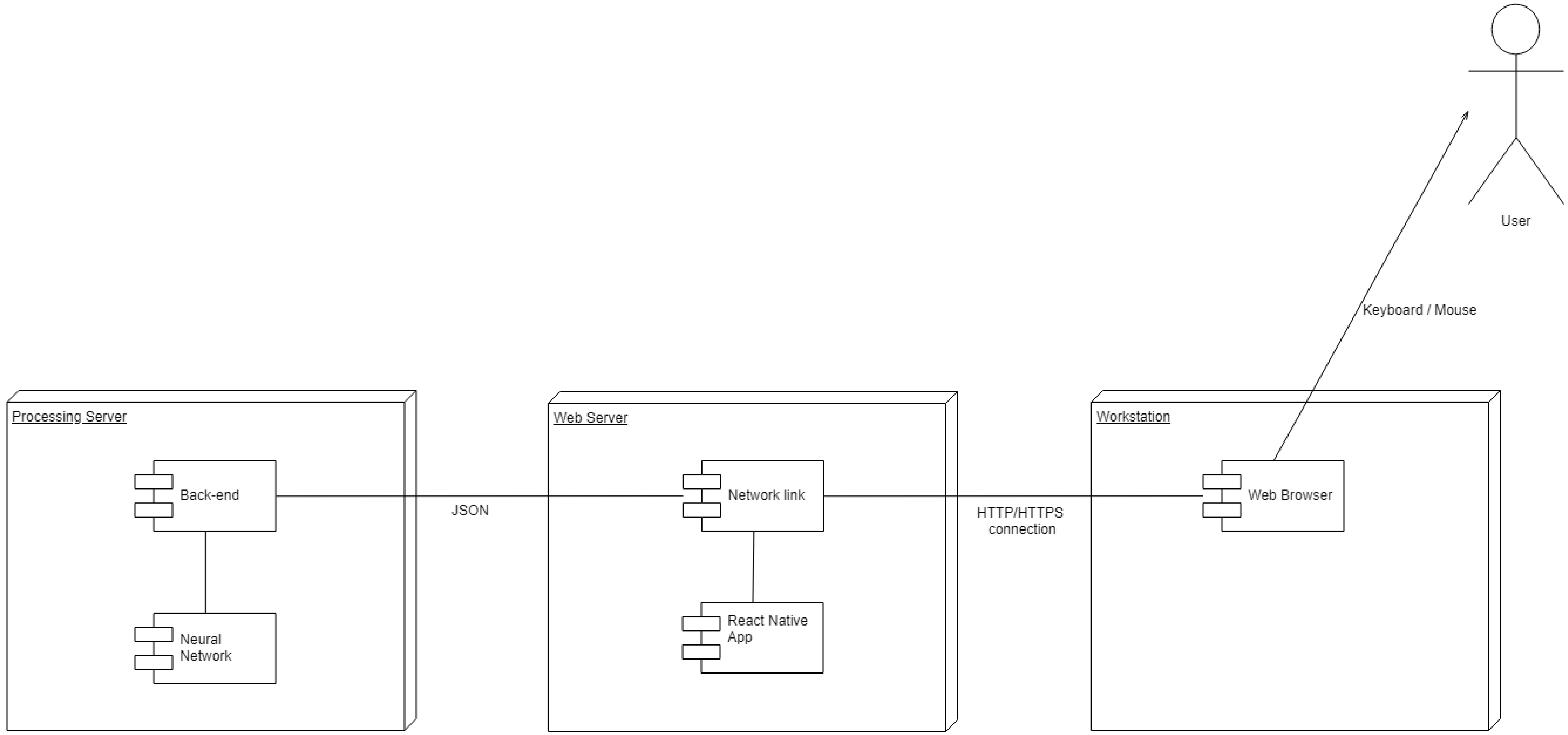


					КПІ.ІП-5106.045490.05.99.Р	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Зарічковий О.А.			Структура генеративної мережі	Літера / Маса / Масштаб
Перевірив		Ісаченко Г.В.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
Затвердив		Ісаченко Г.В.				



Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-5106.045490.05.99.СС		
Розробив		Зарічковий О.А.			Схема структура класів програмного забезпечення	Літера	Маса	Масштаб
Перевірив		Іванюк Г.В.						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Пшук К.І.			Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51	
Затвердив		Іванюк Г.В.					

КПІ.ІП-5106.045490.05.99.СС



					КПІ.ІП-5106.045490.05.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	Діаграма розгортання програмного забезпечення
Розробив		Зарічковий О.А.			Літера / Маса / Масштаб
Перевірив		Ісаченко Г.В.			Аркуш / Аркушів
Т. кон.					
					Програмні засоби для деблюрінга зображень за допомогою генеративних змагальницьких систем
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
Затвердив		Ісаченко Г.В.			